950-035 Technic

Programmable Systems

Teacher's materials



For Apple® II Computers

LEGO Educational Materials Division
LEGO Australia Pty. Limited
PO Box 639
Lane Cove
NSW 2066

In order to maintain the highest level of support for users of Programmable Systems, it is extremely important that purchasers become registered. Support for the existing pack and news of future developments can then be guaranteed.

This is your customer reference number:

A 000402

Include this in all correspondence with LEGO Educational Materials Division, to ensure that your enquiry is dealt with as efficiently as possible.

PLEASE KEEP THIS CARD IN A SAFE PLACE.

Programmable Systems

Please register me as a user of Programmable Systems

Name _____ Position _____

Institution _____

Address _____

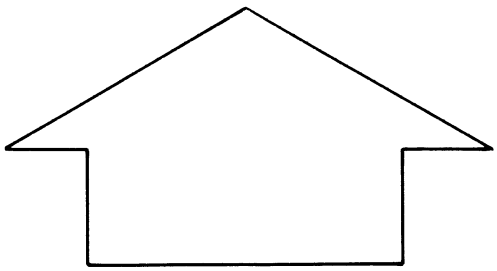
Postcode _____ Tel _____

Were the materials in satisfactory condition on arrival?

☐ Yes ☐ No

Customer reference

A 000402



Detach this card and return it to the address overleaf.

No stamp is required if this is posted in Australia.

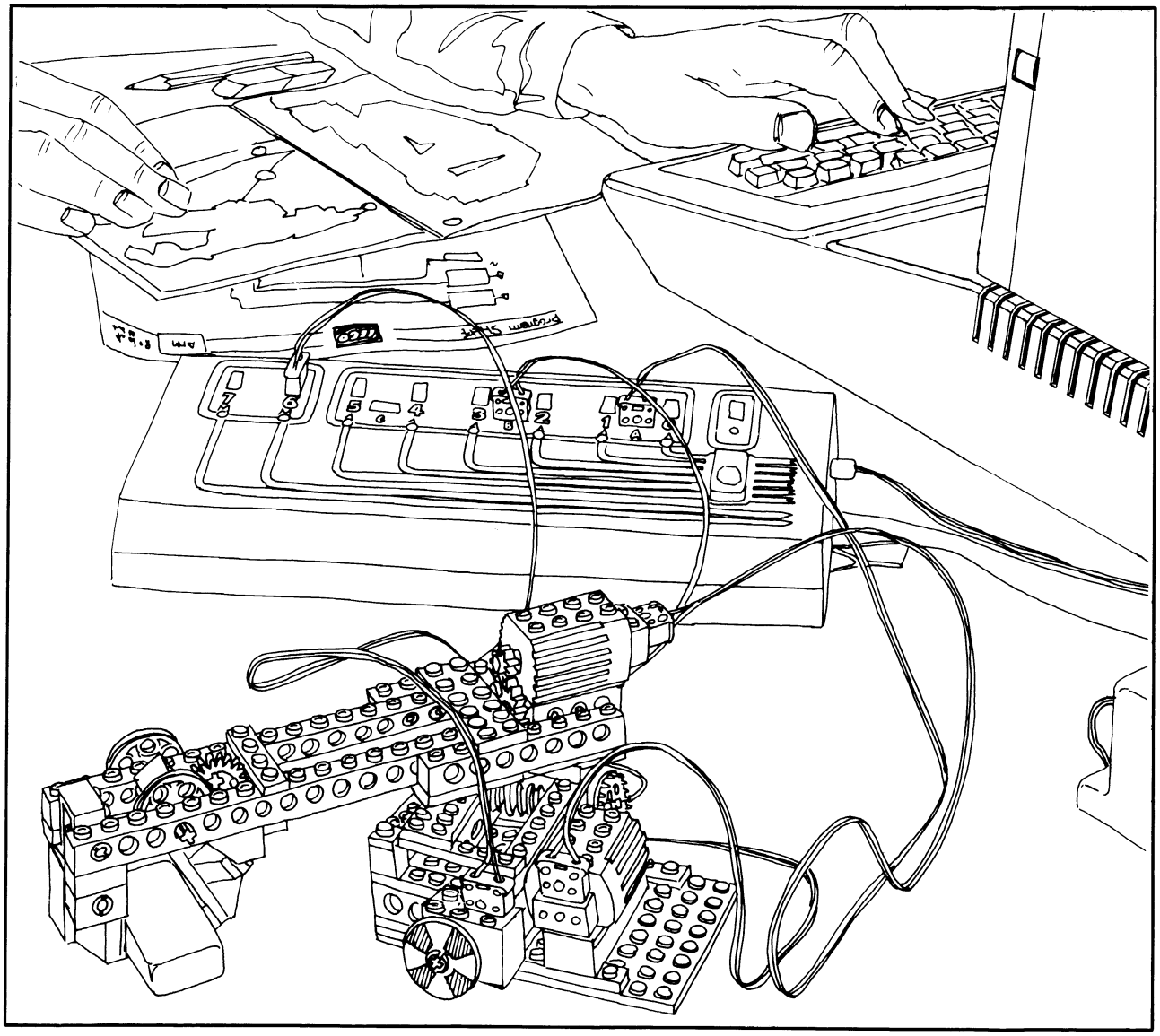
This portion may be discarded.

No postage stamp required
if posted in Australia

BUSINESS REPLY POST
Permit No. 56 — Lane Cove
Postage and fee will be paid on delivery to

LEGO Educational Materials Division
LEGO Australia Pty. Limited
PO Box 639, Lane Cove, 2066, NSW

Programmable Systems



Acknowledgements

Programmable Systems was developed as a result of a collaborative project between The LEGO Group, The Microelectronics Education Programme and British School Technology — Trent.

The development team

Alan Anov

Mike Bostock
Anthony Lucas
Mike Sharpe
Bob Windsor

Andrew Cooper
Alan Giles
Steve Rogers

Many people have made valuable contributions to the educational discussions which have shaped the course of this development and particular thanks are owed to the following:

Graham Bevis, MEP
Barry Evans, Morecombe High School, Lancs
Richard Fothergill, MEP
Roger Holmes, National Centre for School Technology
Mike Ive, HMI
Pauline Jay
Oliver Linton, Harrogate Grammar School, N Yorks
Dr John Martin, Salford University
Geoff Shillito, British School Technology — Trent
Noel Whalley, MEP

For APPLE Computers

The APPLE version development team:

Programmers — Mark Simon and Gregory Simon
Manual amendments — Sue Simon and Peter Smith

Trials team

Australian schools and teachers:

Specifically: Auburn Primary School and Alan Cory,
Patrician Brothers College and Jenny Rickard
Picton High School and Kevin Garrick
Telepea Primary School and Wendy Wakelam

Other contributors:

APPLE Computer Australia Pty Ltd
especially Diana Ryall and Conrad Silvester
LEGO Australia Pty Ltd
especially Diane Barry and Peter Smith
The LEGO Group, Denmark
especially Henning Barnkob

Designed and produced by:

Tecmedia Limited
Loughborough
Leicestershire
England
Tel 0509 230248

ISBN 1 7316 0550 0 Programmable Systems (Whole Pack)
ISBN 1 7316 0722 8 Programmable Systems (Teachers Guide)

® LEGO is a registered trade mark
© 1987 LEGO Group

These materials, both text and computer software, are fully protected by international copyright. If you make a copy and allow anyone else to retain it (***even if you give it to them***), you are committing an offence and are liable to prosecution.

You are advised, however, that the software and many of the materials are intended as photocopy masters and may be freely copied ***for use in the purchasing institution***.

Contents

Section 1: Introduction

Overview 1.1
Resources available 1.2

The resource

Introduction to the resource 1.3
- Main aims of the materials 1.3
- Methodology 1.4
- Assignments 1.5
- Assessment 1.6
- Management of materials — some questions and answers 1.7

The materials

Introduction to the materials 1.8
- Hardware 1.8
- Software 1.8
- Teacher's guide to the assignments 1.9
- The assignments 1.10
- Student's resources booklet 1.10

Hands-on practice 1.11
- Preparation 1.11

Using LEGO *Lines* 1.12

LEGO *Lines* User Guide 1.16
- What is *Lines* about? 1.16
- How does *Lines* work? 1.16
- Understanding the screen 1.16
- Creating a *Lines* program 1.17
- *Lines* utilities 1.21
- Error messages 1.23

LEGO *Lines:* program design 1.24
- Key software techniques 1.26
- Some program builders' hints 1.26
- Structured flow diagrams 1.28
- Adapting or combining models 1.29

LEGO *Lines:* sample programs 1.31

Section 2: Classroom materials

Guide to equipment 2.1
Guide to resources 2.2
Guide to assignments 2.4

Contents *(continued)*

The assignments

Assignment 0: Introduction to the materials 2.8
Assignment 1: Introduction to programmable systems 2.12
Assignment 2: Introduction to computer control 2.16
Assignment 3: Introduction to manual control 2.28
Assignment 4: Sensing and feedback 2.30
Assignment 5: Computer control 2.34
Assignment 6: Manual control 2.46
Assignment 7: Solving a problem 2.48
Assignment 8: Designing a structured program 2.50
Assignment 9: Designing a programmed system 2.52

Section 3: Resources booklet

Guide to contents 3.3
Organising yourself to solve problems 3.5
Using the equipment 3.10
Designing control programs using the LEGO materials 3.20
Glossary of terms 3.31

Section 4: Technical reference

Hardware 4.1
Setting up your computer 4.1
Using an Apple® IIGS 4.2
Software 4.3
Preparing disks 4.3
Using LEGO ***Lines*** 4.4
Starting LEGO ***Lines*** 4.4
Terminology 4.4
The Apple LEGO ***Lines*** screen 4.5
Entering data 4.5
Special functions 4.6
Editing functions 4.8
Info functions 4.9
Test and Run functions 4.9
Disk functions 4.9
View functions 4.11
Print function 4.11
Miscellaneous functions 4.11
Exit function 4.12

Contents *(continued)*

Programming the LEGO Interface 4.13
- The Slot Card 4.13
- Programming in AppleSoft® 4.13
- Reading data 4.14
- Sending data 4.15
- Programming in machine language 4.16
- Programming in LOGO II 4.18
- A closing note 4.19

Section 5: Duplication masters

Program sheet 5.1

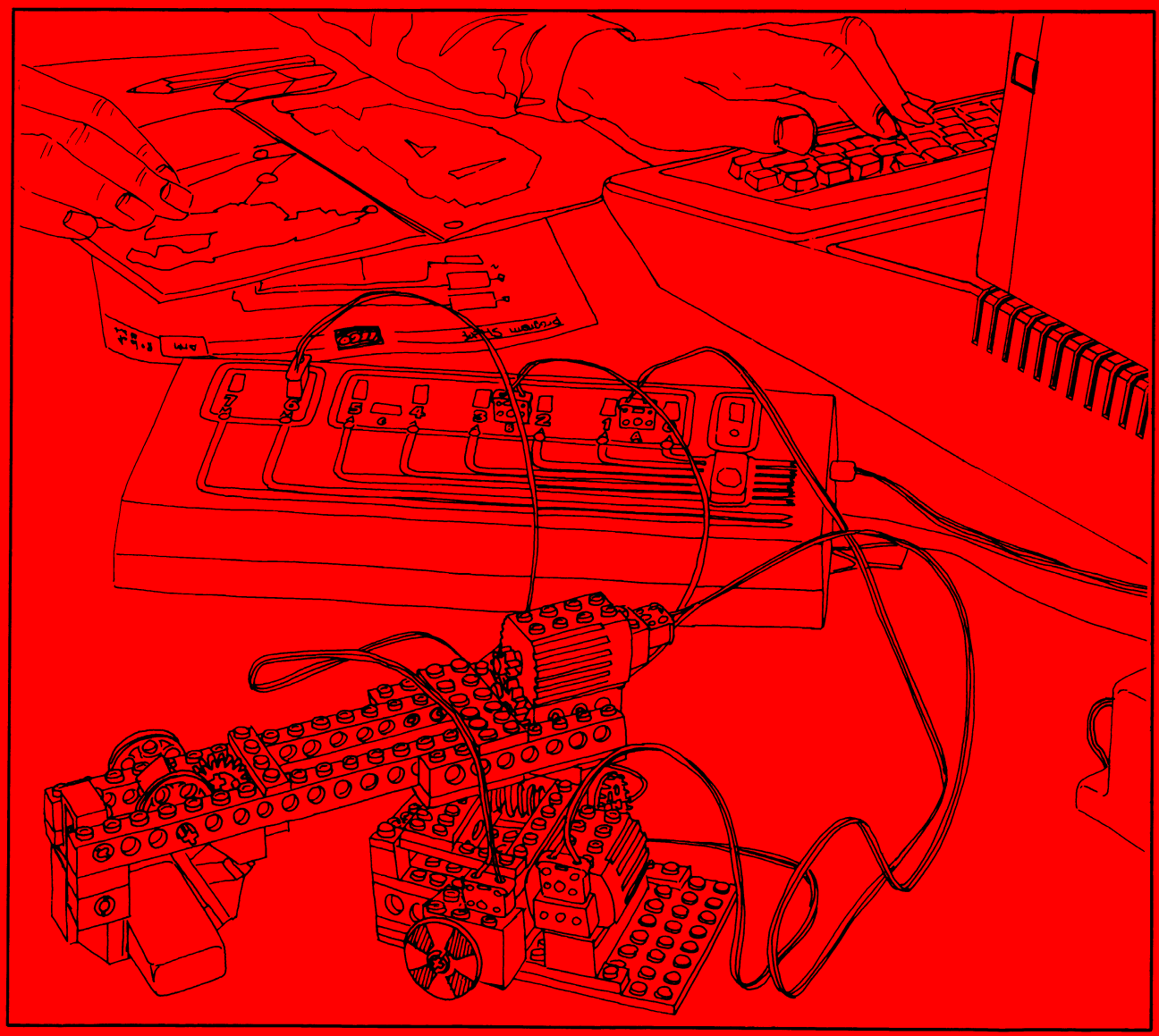
Assessment sheet 5.2

Manual controller keystrips 5.4

Section 1

Introduction

Programmable Systems



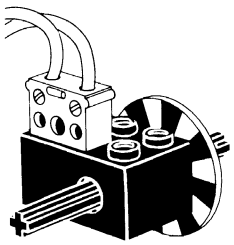
A resource of hardware and software which develops understanding of Technology in a challenging way

Overview

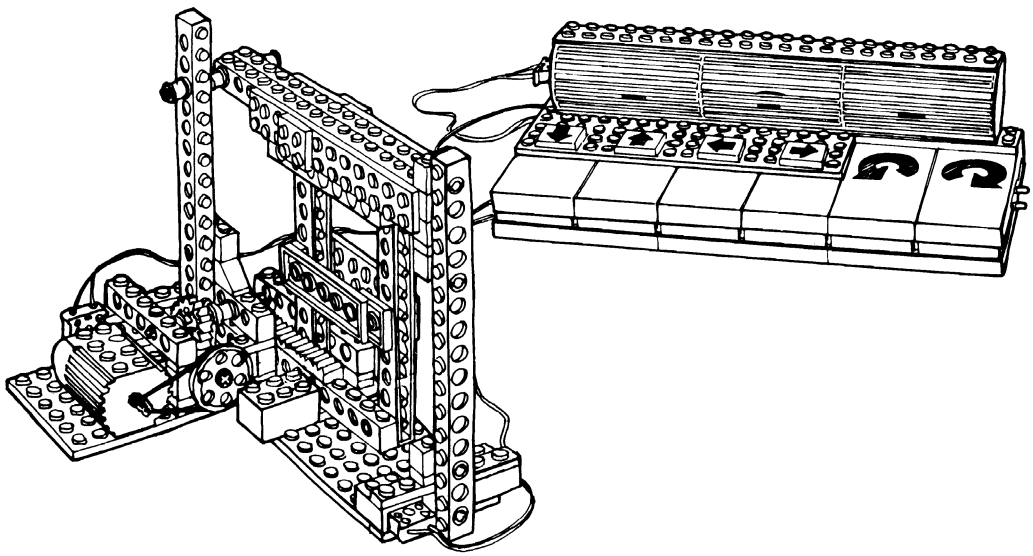
The aim of this material is to develop a student's competence in solving problems.

Students will be placed in learning situations which pose practical problems in an enjoyable way. This provides a progressive and effective learning strategy enabling students to practise technological skills valued in many walks of life.

Many students do not understand problem-solving nor do they appreciate microprocessor control yet this technology is all around us. These materials provide a framework within which to ***come to terms with*** important technological processes, to ***understand*** them and, most important, to ***apply*** them.



Students need confidence to approach problems and solve them successfully. They learn that by applying a combination of techniques, logic and imaginative ideas they can arrive at a sound and effective solution. Working within a team they learn to work co-operatively and to communicate their ideas. They learn to be critical of what they do through a process of self-assessment. They begin to understand some of the fundamental concepts of technological control.



The materials are designed primarily for those in the 11–14 age group but are suitable for older students. They may be used in courses in Technology: Computer Studies, Control Technology, CDT, Science, Integrated Studies and General Studies.

Before embarking on a structured approach to the LEGO Technic Control materials the children should have ample opportunity to explore LEGO Technic Sets I and II.

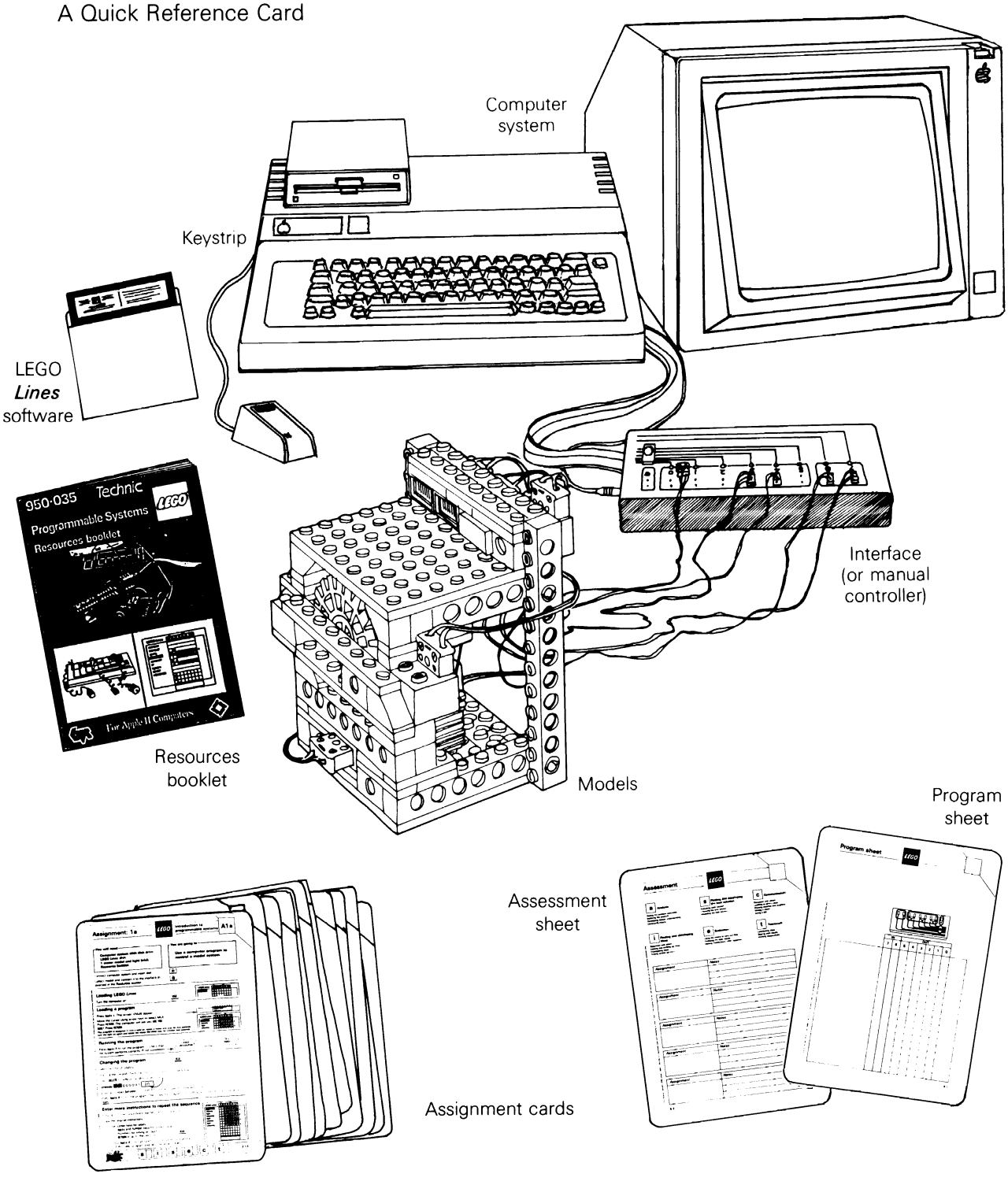
LEGO materials are not 'easy', particularly LEGO Technic materials and all students and teachers should become well acquainted with the techniques needed to manipulate them.

Resources available

This kit consists of:

- A Teacher's Materials Book
- A set of Student Resource Booklets
- A set of Student Assignments
- A LEGO ***Lines*** Disk
- A Quick Reference Card

The assignments form a coherent course in **informatics** but the materials as a whole form a substantial resource about technology and control.



Introduction to the resource

This resource pack is suited to teachers who are new to the subject of control technology and those who, experienced in the field, are looking for effective materials to work with. Little pre-knowledge has been assumed by the authors, only some familiarity with the LEGO materials and the elements of the computer hardware. It is expected, however, that the teachers will familiarise themselves thoroughly with the materials before using them in the classroom. There is substantial support and introductory material to help teachers achieve this.

Main aims of the materials

- To provide a practical experience in the design of programmable systems using hardware and software in the form of rapidly-adaptable functional units.
- To develop an understanding of the characteristics of microelectronic systems, from binary operations to closed-loop control.
- To study the role of the microcomputer as an example of a programmable, decision-making, switchable device.
- To encourage systems thinking as it applies to the design cycle.
- To promote problem-solving activities in the context of recognisable examples of technological systems.

Controlling devices using a microelectronic system, computer or other, is an area of the curriculum which offers a great many educational opportunities. Students who are able to use this resource gain access to many fundamental technological concepts and processes.

Central to these experiences is the design process:

— Analysing the problem
— Researching alternative (but possible) solutions
— Refining ideas
— Selection of the most appropriate solution
— Building (hardware and software)
— Testing that solution
— Evaluating the success of that solution

and finally,

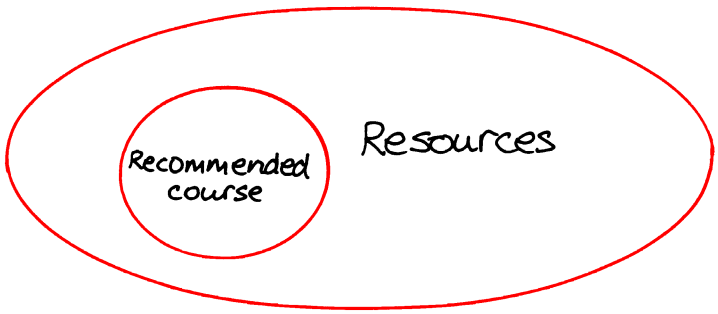
— Communicating the process of arriving at that solution to others.

Prominent also is the ***systems approach***, the capability of seeing the overall purpose, the holistic view, while paying attention to the detail, of understanding the interdependency and nature of the separate elements which interact to form a complete system.

Students involved in the design and control of such systems will play many roles. As they engage in the structural, mechanical and logical aspects they will be engineers, systems analysts and programmers. Each role will contribute its particular insight into some of the processes which characterise the world we live in.

Methodology

The materials as a whole should be regarded as a flexible resource. This has many implications for the way it should be used, from both the teacher's and the students' point of view. The authors have assumed that each teacher will know better than anybody else how to get the best results from the materials within his or her own context.



A recommended set of assignments is provided, together with a ***Resources booklet*** for the students. Once the effective use of this booklet is established, teachers should be released from the task of providing information on the techniques, concepts and practical detail of using the materials. In this way they can devote more attention to the management of the learning process, to the way in which learning takes place within the class, in the groups and for the individual student.

In the assignments, there is substantial cross-referencing to the ***Resources booklet***. This has been designed so that extracting information is as straightforward as possible; a ***Guide to contents*** and a ***Glossary of terms*** complement the actual resource pages on approaching problems, using the equipment and designing control programs. Once students can use this booklet with confidence, they can take far more responsibility for their own learning process. They will work at a pace most suited to their individual needs and should begin to utilise the materials in more constructive and inventive ways.

Such a process of student learning inevitably places a responsibility on teachers to develop their own strategy to exploit these materials. This will be concerned with how to manage the resource to the best advantage. Decisions can only be taken once the range and scope of the materials are understood. To support this understanding, an ***Introduction to the materials*** section has been included (pages 1.8 to 1.10).

As the understanding of the resource develops, teachers may wish to extend its scope, perhaps by adding articles or particular information sheets to the Resources booklet.

Further building materials are available in the form of LEGO Technic 1092 sets, or even the use of other materials with existing LEGO bricks. Ideas for this type of activity are included in the ***Teacher's guides*** to LEGO Technic 1 and Technic 2.

The LEGO ***Lines*** program is a software control system for a number of external devices supplied by the LEGO Group. A series of routines, written in Applesoft BASIC, Logo and Machine Language, which enable you to emulate some of the ways in which ***Lines*** controls the LEGO Interface, is included in the ***Technical reference*** section, page 4.13. This could form the basis of further work, particularly with the LEGO 1092 set.

Assignments

A number of student assignments are provided in the resource:

Assignment 0 — **A0 (a-c)**
Assignment 1 — **A1 (a-b)**
Assignment 2 — **A2 (a-f)**
Assignment 3 — **A3**
Assignment 4 — **A4 (a-b)**
Assignment 5 — **A5 (a-f)**
Assignment 6 — **A6**
Assignment 7 — **A7**
Assignment 8 — **A8**
Assignment 9 — **A9 (a-j)**

Assignment **A0** is intended as an introduction to the materials for the students and enough time should be allocated to ensure all students attempt the questions found there. All the assignments (**A1, A2, A7** and so on) are designed to be completed in a minimum of a double lesson. There is, however, a lot of material in each sheet and the recommended course could occupy a great deal longer than the time suggested.

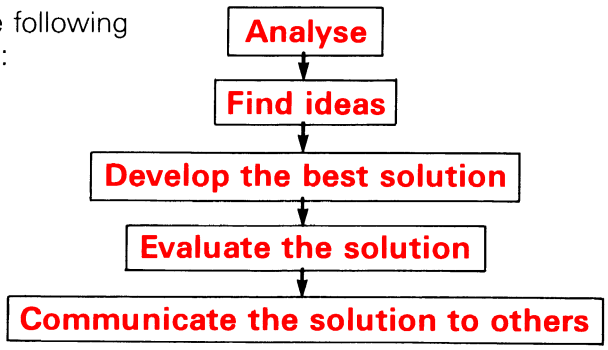
Although the assignments do appear in order, it is up to individual teachers to discover the most appropriate schedule and route through them, as dictated by their own situation.

Assignments have been combined to cater for a situation where there are five computer systems available but more than fifteen students, providing a workable mix of mechanical and computer control activities. These are grouped as follows:

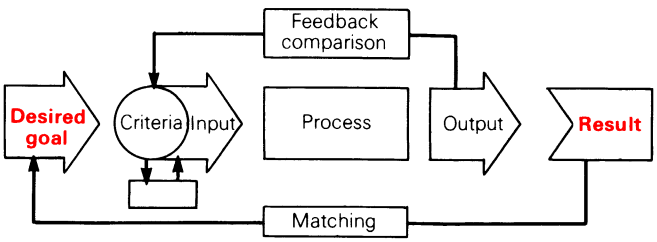
Assignment **A1a** with Assignment **A1b**
Assignment **A2 (a-f)** with Assignment **A3**
Assignment **A4a** with Assignment **A4b**
Assignment **A5 (a-f)** with Assignment **A6**

Assignments **A6**, **A7** and **A8** are central to the course in that, by the time these are encountered (if the assignments are being followed in order), the students will need the structured approach to problem-solving presented to them.

The authors have used the following model for problem-solving:

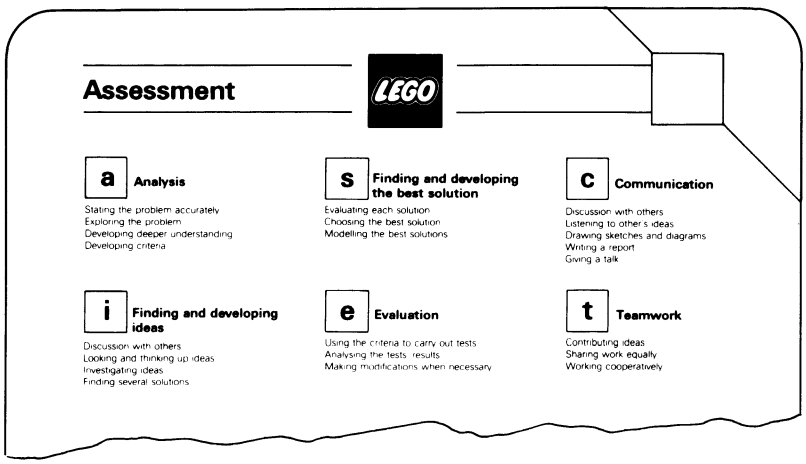


In reality, the technological process is far more complex than this and it is expected that the students will demand more of this simple model as their understanding grows.

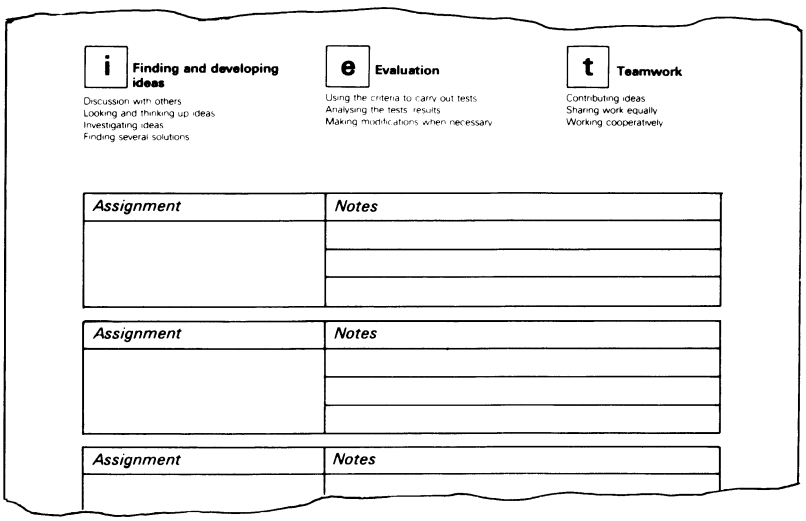


Assessment

A process of continuous negotiated assessment is recommended. This is linked to the use of the boxed characters on the Assessment sheet, descriptors of the main elements of the problem-solving model outlined previously (page 1.5).



The boxed characters are repeated on the bottom of each assignment card (except **A0**). They are intended as a reminder to the students, providing a framework in which to assess their own progress. The teacher is also expected to assess group and individual performance and it is anticipated that, at some agreed point, students and teacher will get together to negotiate the overall assessment. This process has to be facilitated by the teacher and should occur in whichever way is most manageable in the particular context. The Assessment sheet provides space for students to write in a description of how they have performed on the assignment concerned.



Students will soon appreciate that not all skills are required for all activities. The teacher will need to support this understanding and time should be made available during the course of each assignment to discuss whether or not a skill has been used. A formative mechanism of assessment is, therefore, being employed.

Management of materials — some questions and answers

— *What is the best physical location for the course (assuming a group of 30 students)?*

A shared computer area where there is desktop and demonstration space.

— *What is the best group size?*

Probably only two, but most likely three students will be the working group size, especially where the class size approaches thirty students.

— *How will I, a teacher, become sufficiently familiar with the materials?*

By working carefully and slowly through the ***Introduction to the materials*** section and the ***hands-on practice***.

— *What is the best way of organising the recommended course?*

Once you are familiar with the materials, you will use them as you think fit. The likelihood is that, first time round, you will follow the recommended course as presented in the Assignment cards. As you become more confident in their use, you will re-order and add new materials of your own.

— *What kind of computer system is required?*

Any Apple® IIe or Apple® IIGS computer with at least one disk drive and 128K of memory. The use of an AppleMouse® is optional, but recommended.

Additional hardware, such as a printer or second disk drive is optional.

— *What is the best way of organising time?*

Each assignment is designed to take the equivalent double period but this, of course, is only a guide. You may find that your students take half the time, or twice as long. It is, however, recommended that you look to increase the time available for each assignment, to exploit the resource materials and to organise the assessment scheme.

— *How can the recommended assessment mechanism be employed?*

The assignment sheets contain guidelines for student self-assessment which may be incorporated into your own assessment procedure. Each assignment also has clearly stated objectives which may be assessed.

— *How should the students be introduced to the materials?*

It is recommended that all students work carefully and slowly through Assessment **A0 (a-c)** which has been specially written for this purpose.

— *Where do the materials lead?*

They generate a lot of enthusiasm on the part of the students. The resources may be used to explore other models or may be extended by other LEGO Technic sets, particularly 1092 or LEGO Technic 1 and 2.

Software may be developed by using routines similar to those included in the ***Technical reference*** section, page 4.13.

The method and content lead naturally to further courses in control technology and other cross-curricula activities.

Introduction to the materials

The success of these materials depends to a large extent on the individual teacher's familiarity and confidence with them.

This is an introductory section which teachers work through prior to presenting the materials in the classroom.

It takes you through:

— the hardware — the LEGO sets and the computer system
— the software — LEGO ***Lines*** and creating working masters
— the teacher's guide to the assignments
— the assignments
— the students' resources
— some hands-on practice.

Hardware

LEGO sets

A guide to the LEGO sets, entitled ***Directions for use***, can be found in the carton in which they arrive. This is accompanied by the series of construction guides which give an idea of the models which can be built.

Computer system

LEGO ***Lines*** requires the Apple® IIe computer system to have at least one disk drive and 128K RAM. A mouse is optional, however it is recommended. You may also want to have a printer attached. It is assumed that the interfaces are connected as follows:

Slot 1: Printer Interface (Optional)
Slot 2: LEGO Slot Card
Slot 4: AppleMouse® Interface (Optional)
Slot 6: Disk Interface (see Resources Booklet pages 3–11)

Software

LEGO Lines

The software is supplied on a master disk. Either 5.25" or 3.5" disks can be used to back up this program.

Data disks must be initialised prior to use. The ProDOS User's Disk is quite adequate for this task.

The ***Technical reference*** section, page 4.3 contains descriptions of the software on the disk.

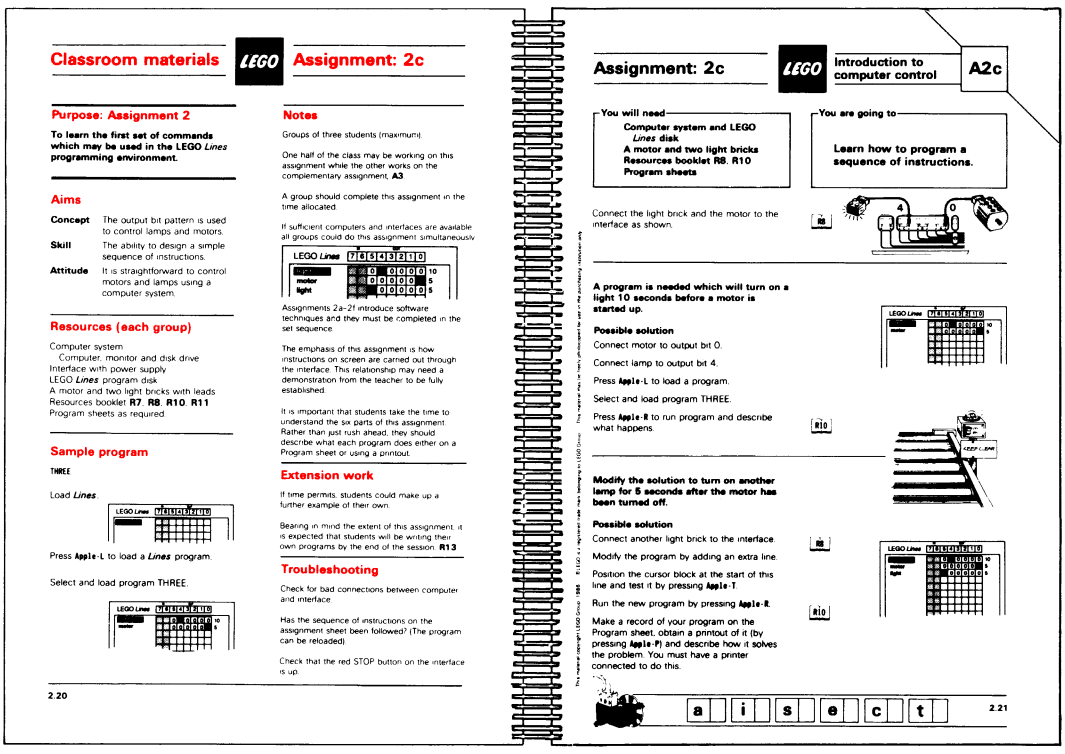
As soon as you receive this pack, BACK UP BOTH SIDES OF YOUR DISK (as explained in the ***Technical reference*** section, page 4.3). Keep the original in a safe place and use the copies as working masters. The Demonstration Programs are described in more detail in the ***LEGO Lines: Sample programs*** section (pages 1.31 to 1.41).

When groups are working on assignments involving LEGO ***Lines***, they will require a working master disk, which will be a copy of your working master. In addition to the LEGO ***Lines*** software, they should have the appropriate sample program (indicated on the teacher's commentary page accompanying each assignment) loaded on the same disk. This will involve housekeeping to ensure that the working masters, are kept up to date with the assignments being worked on. A detailed description of making backup copies, maintaining working masters and housekeeping procedures is given in the ***Technical reference*** section, page 4.3. There should always be room on this working disk for the group to save its own programs.

Teacher's guide to the assignments

The assignments are to be found in the ***Classroom materials*** section which follows. The assignment cards are duplicated here, faced by a commentary designed for the teacher. Where an assignment is split into many parts, Assignment **A2 (a-f)** for example, this information is duplicated opposite each part, with minor differences relevant only to that part.

Each commentary states the purpose of that assignment, the concepts, skills and attitudes it promotes and lists of equipment and sample programs required. Notes and pointers on the assignment follow, together with ideas for extension work and a troubleshooting section. Often, the assignment itself requires the modification of one of the sample programs in order to solve a particular problem. Where this is the case, the modified program is given in the teacher's commentary for reference.



Classroom materials — Assignment: 2c

Purpose: Assignment 2

To learn the first set of commands which may be used in the LEGO *Lines* programming environment.

Aims

Concept The output bit pattern is used to control lamps and motors.

Skill The ability to design a simple sequence of instructions.

Attitude It is straightforward to control motors and lamps using a computer system.

Resources (each group)

Computer system
Computer, monitor and disk drive
Interface with power supply
LEGO *Lines* program disk
A motor and two light bricks with leads
Resources booklet **R7**, **R8**, **R10**, **R11**
Program sheets as required

Sample program

THREE

Load *Lines*.

Press **Apple-L** to load a *Lines* program.

Select and load program THREE.

Notes

Groups of three students (maximum).

One half of the class may be working on this assignment while the other works on the complementary assignment, **A3**.

A group should complete this assignment in the time allocated.

If sufficient computers and interfaces are available all groups could do this assignment simultaneously.

Assignments 2a–2f introduce software techniques and they must be completed in the set sequence.

The emphasis of this assignment is how instructions on screen are carried out through the interface. This relationship may need a demonstration from the teacher to be fully established.

It is important that students take the time to understand the six parts of this assignment. Rather than just rush ahead, they should describe what each program does either on a Program sheet or using a printout.

Extension work

If time permits, students could make up a further example of their own.

Bearing in mind the extent of this assignment, it is expected that students will be writing their own programs by the end of the session. **R13**

Troubleshooting

Check for bad connections between computer and interface.

Has the sequence of instructions on the assignment sheet been followed? (The program can be reloaded).

Check that the red STOP button on the interface is up.

2.20

Assignment: 2c — Introduction to computer control — A2c

You will need
Computer system and LEGO *Lines* disk
A motor and two light bricks
Resources booklet R8, R10
Program sheets

You are going to
Learn how to program a sequence of instructions.

Connect the light brick and the motor to the interface as shown.

A program is needed which will turn on a light 10 seconds before a motor is started up.

Possible solution

Connect motor to output bit 0.

Connect lamp to output bit 4.

Press **Apple-L** to load a program.

Select and load program THREE.

Press **Apple-R** to run program and describe what happens.

Modify the solution to turn on another lamp for 5 seconds after the motor has been turned off.

Possible solution

Connect another light brick to the interface.

Modify the program by adding an extra line.

Position the cursor block at the start of this line and test it by pressing **Apple-T**.

Run the new program by pressing **Apple-R**.

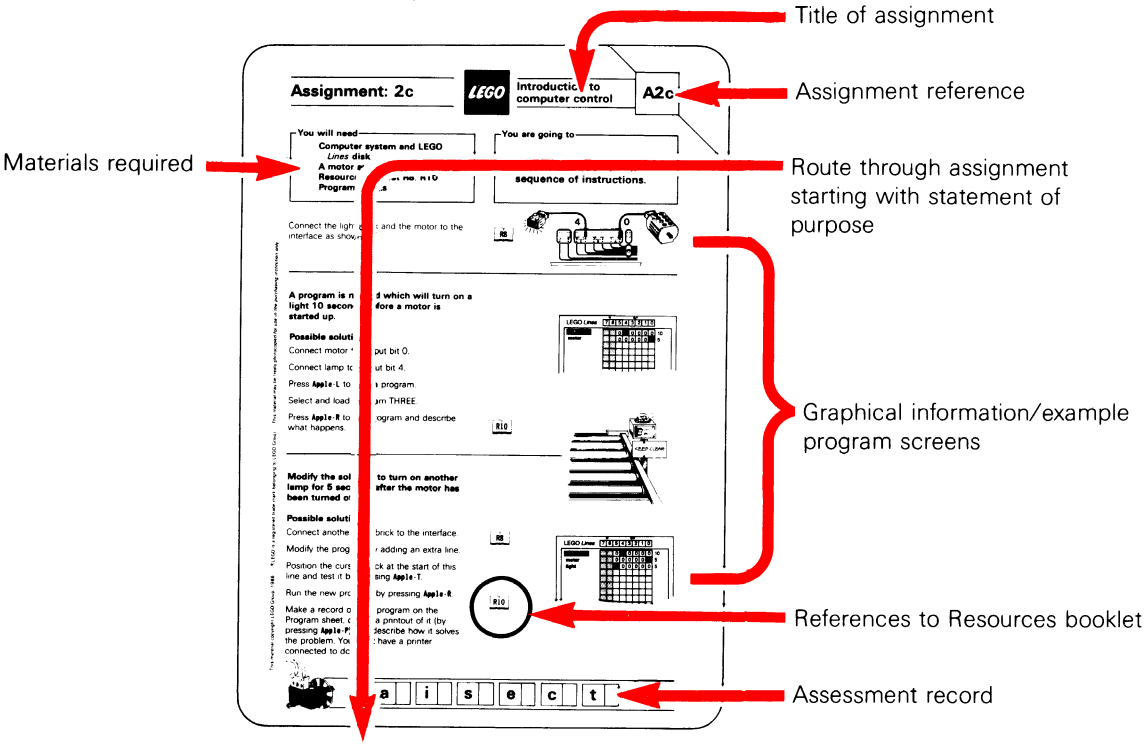
Make a record of your program on the Program sheet, obtain a printout of it (by pressing **Apple-P**) and describe how it solves the problem. You must have a printer connected to do this.

a i s e c t

2.21

The assignments

The layout of each card is intended to make the assignments as easy to follow as possible. Instructions for the student are clearly visible, together with the supporting information and reference to the ***Resources booklet***. The aim of the assignment (or part of an assignment) and the equipment needed to achieve it is clearly stated.

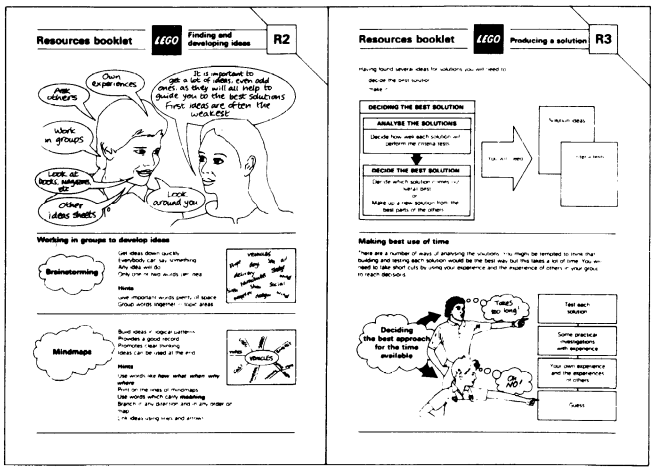


Students' resources booklet

This is an open resource which students can refer to for facts, ideas, techniques and background material. It is organised into three sections:

Organising yourself to solve problems
Using the equipment
Designing control programs using the LEGO materials

It is worth investigating this booklet thoroughly before using the materials in the classroom.



Hands-on practice

This section will familiarise the teacher with the scope of the materials available. It is recommended that the teacher spends time going through these activities in some detail.

Preparation

Computer system

1 Set up the computer as explained in the ***Technical reference*** section, page 4.1.

2 Prepare a copy of the LEGO ***Lines*** master disk. This exercise will make use of the sample ***Lines*** programs, ATEST, BTEST and CTEST, as described in the ***Technical reference*** section, page 4.3.

Models

Construct the following models:

1 ***Automatic door*** (construction guide 1090 B)

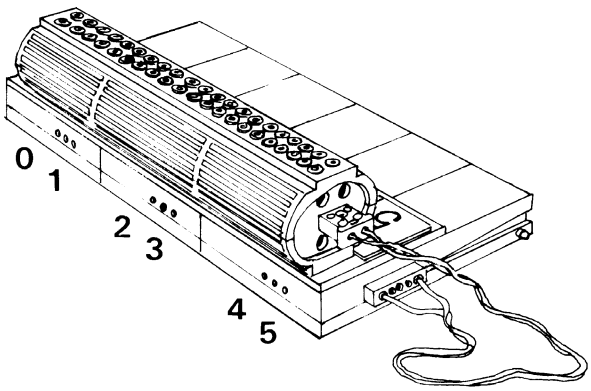
If you only have one 1090 set and no other Technic materials, do not build the models below until you have finished the activities using the sliding door (pages 1.11 to 1.15).

2 ***Gearbox*** (Assignment **A3** and Resources booklet **R18**)

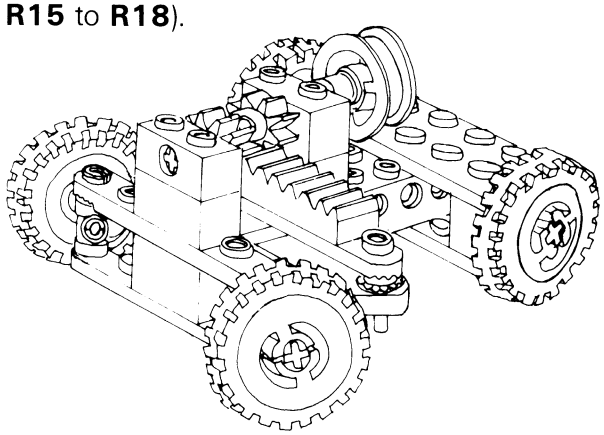
3 ***Vehicle*** (Assignment **A6** and Resources booklet **R15** to **R18**)

Familiarisation

Work through Assignment **A1b** in Resources booklet **R6**, using the Automatic Door model instead of the two-motor model suggested there. This will familiarise you with the LEGO manual controller 1039.

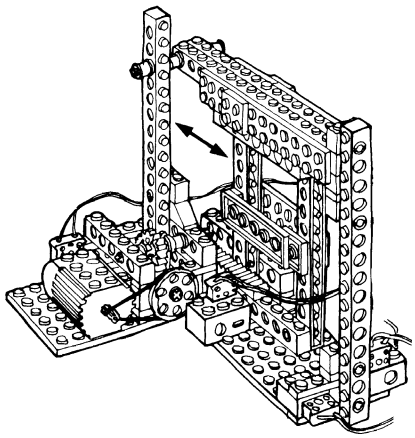


Further practice with the manual controller can be gained with the Vehicle model (Assignment **A6** and Resources booklet **R15** to **R18**).



Using LEGO *Lines*

This section uses only keyboard facilities to operate the LEGO ***Lines*** programs, however the Mouse can be used. Full details of Mouse operation can be found in the ***Technical reference*** section, page 4.4.

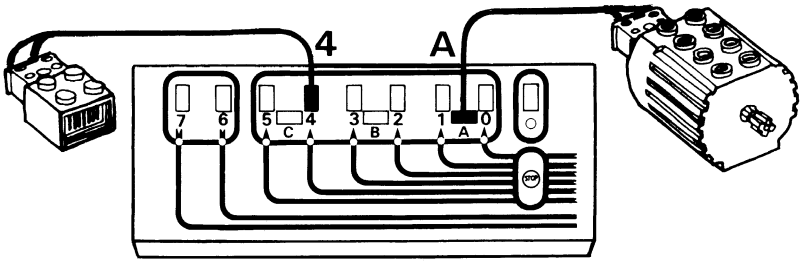


1 Connect the computer interface

Use the Automatic door model, starting with the door closed.

Connect the motor and one of the lights to the interface. The light should be connected to output socket 4, and the motor should be connected between output sockets 0 and 1. (See diagram below. Make sure that the red STOP button is up.) (**R8**)

If the button is pressed down, all the lights will go out, with exception of the continuous power output — marked ***4v***.

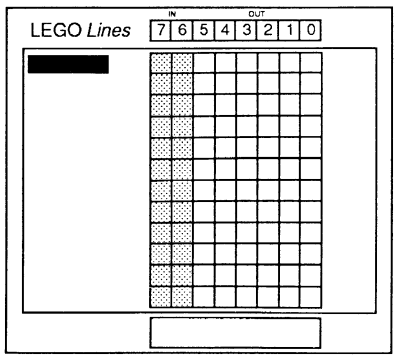


2 Start the *Lines* program

Start the LEGO ***Lines*** program disk, by placing the disk in the disk drive and turning the computer on. After a few moments the program screen will appear.

In describing how to use the LEGO ***Lines*** program, the following terms and notation will be used:

Using keys:	`Press`	Press the key, and then release it.
	`Hold`	Press the key and keep holding it down.
	`Apple-Key`	Hold the ⌘ key. Press the second key, then release it. Release the ⌘ key (for example, ⌘ -`X`).

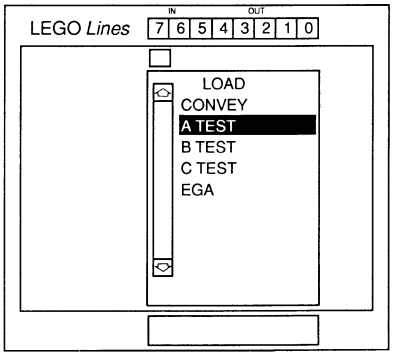


3 Open and close the door

Turn bit 0 on (**Apple-0**). You will find that a 1 has appeared in that position on the display. Now run the program (**Apple-R**).

The door should open or attempt to close. If the model attempts to close, then the plugs on the interface should be reversed. If nothing happens at all, then check the connections and try again (**R8**).

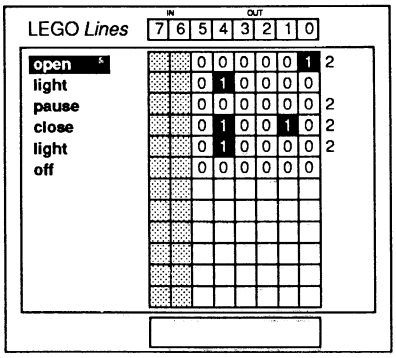
Now turn bit 0 off (**Apple-0** again) and turn bit 1 on (**Apple-1**). This should now close the door.



4 Load and run ATEST

Load the ***Lines*** program ATEST from your disk. To do this press **Apple-L**. A list of sample programs will appear on the screen. Move the cursor to the desired program, ATEST, using the arrow keys press **RETURN**. The program will then load. This will take a few moments while the program is being read into your computer's memory. When the program has finished loading, run it (**Apple-R**).

ATEST is designed to open the door and light a lamp; after a pause, the door will close, and the lamp will light again.



As the program is running, the line being executed will be highlighted on the display screen. At the same time, the lights on the interface indicate which bits are being turned on and off.

5 Changing the program

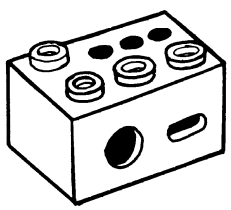
You may move the cursor block to a different line by pressing the ↑ or ↓ key. Use these keys to move to one of the lines labelled: **light**. Write down on a piece of paper the bit settings and timing.

Use the **DELETE** command to delete the line (**Apple-D**). Notice that the lines below move up to fill the gap.

Run the program again (**Apple-R**), and observe what happens to the light.

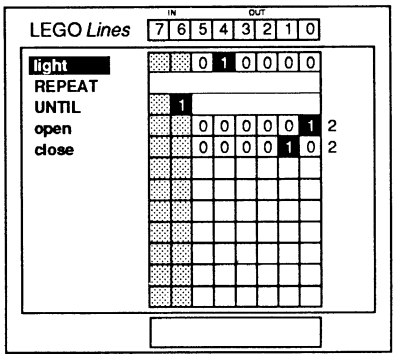
6 The opto-sensor

Find an opto-sensor brick. This is a small black brick with holes in the side to let light in. Now use a lead to connect the opto-sensor to input bit 6 on the interface.



The opto-sensor is used to detect changes in the light intensity entering it. Move your finger on and off the opening, and notice what happens to the green light on the interface (**R9**).

7 Load and run BTEST



Load the ***Lines*** program BTEST from your disk as you did in step 4 (when you loaded ATEST). Write down on a piece of paper the bit settings and number.

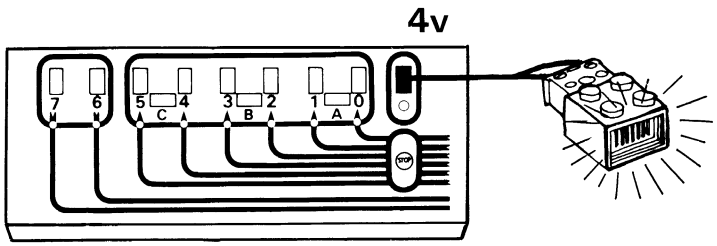
Before you run this program, make sure that the light on bit 6 of the interface is OFF, either by putting your finger on the opening, or by taking it off.

Run the program (**Apple-R**). Nothing should happen to the door of the model. Now make sure that the indicator light on bit 6 of the interface is ON, either by taking your finger off the opto-sensor's opening, or by putting it on. This time, the door of the model should open and close.

8 Resetting the opto-sensor

The state of the opto-sensor is said to ***float***. This means that it could be on or off initially. It is usually desirable to have it in a known state before using it.

Use a lead to connect a light brick to the 4v output on the interface (above the STOP button). This light may now be used to reset the opto-sensor, by bringing the light close to the opto-sensor.



9 Adding a line

We will add a line to the end of the program to turn a light on for three seconds. To do this, use the ↓ key to move the cursor to the end of the program.

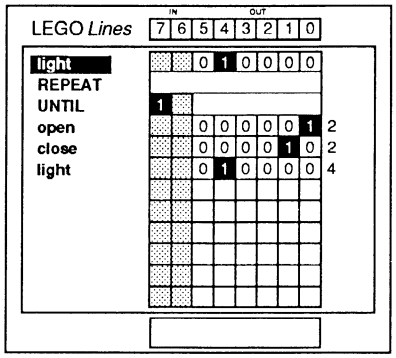
Type the letters: **l i g h t**. The word will appear on the left column, in lower case. This is a label.

Type the number **3**. This will appear in the right column, and is a value. In this case, it is the number of seconds the light will stay on.

Now turn bit 4 on (**Apple-4**). This will operate the light connected to bit 4 on the interface.

Now test this line (**Apple-T**). This command only runs the line where the cursor is positioned. It is also a useful command for testing connections to your interface.

10 Changing lines



It is possible to change existing lines in the program. To do this, use the ↑ or ↓ key to move the cursor to the desired line.

You can alter the bit values of a line. In this example, we will change the **UNTIL** line, to tell the program to check for the opto-sensor in bit 7 rather than in bit 6.

Notice that the value of bit 7 is shaded: this stands for 'Any Value', meaning that the program will not bother checking this bit at all. Set bit 7 to 1. To do this press **Apple-7** and keep 'toggling' key 7 until **1** appears in the box. Then set bit 6 to **any value**. To do this press **Apple-6** and keep 'toggling' key 6 until the box is shaded.

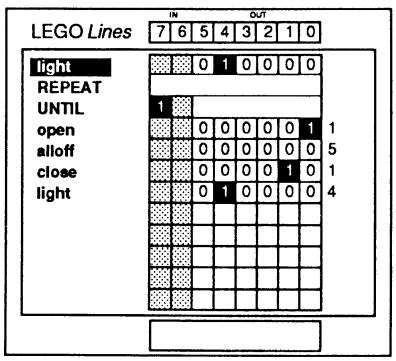
Remove the opto-sensor from bit 6 and connect it to bit 7. Then run the program (**Apple-R**) as before.

You can also alter the values of a line. To do this, use the ↑ or ↓ key to move the cursor to the line labelled 'open'. Then use the →key to move the cursor to the values column.

Now use the **DELETE** key to erase the current value. Then type in a **1**, to open the door for a shorter time.

Change the value of the 'close' command to **1** the same way as above. Then run your program (**Apple-R**).

11 Inserting a line

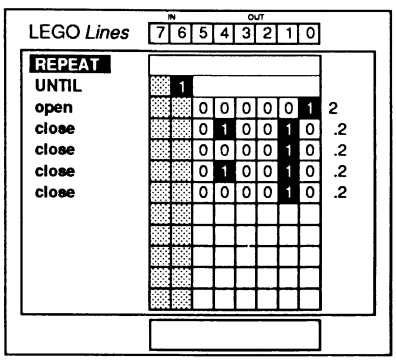


It is possible to insert a new line between existing lines of a program.

Move the cursor to the line labelled 'close', to insert a line before it. Then insert (**Apple-I**). A blank line will appear, pushing the current line, and others below it, down.

The new line will appear blank. Actually, it comes with all output bits set to off (0).

Enter a new command, **alloff**, which will turn everything off for five seconds. To do this, type in the word **alloff**, which will appear in the labels column. Then type the number **5**, which will appear in the values column. Then run the program (**Apple-R**).



Can you make the light flash on and off while the door is closing? (A solution is shown in the program CTEST which uses input bit 6 for the opto-sensor.)

LEGO *Lines* User Guide

What is *Lines* about?

Lines is designed to be an introduction to building a control program.

Control is effected by supplying or denying power to a set of connections on the interface usually switching motors or lamps on and off.

Lines is suitable for students of all abilities from the age of 11 upwards.

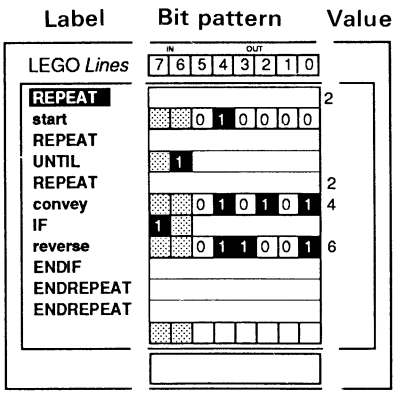
How does *Lines* work?

Lines is a controller which treats a series of instructions as a control program which it can use to control a LEGO model through the LEGO Interface.

Each instruction occupies a single line on screen. The instruction describes a state on the interface or a condition which it expects to find.

Understanding the screen

Lines is designed to accept instructions with a minimum of keystrokes. These instructions are displayed on the screen.



An instruction line has the following structure:

label + **bit pattern** + **value**

Labels

Labels may either be a keyword (a ***Lines*** reserved word) or a word that you make up to describe a bit pattern.

Bit pattern

The bit pattern defines the output state on the condition being tested. Bits 0–5 are the output bits. If there is a 1 in the bit it is said to be on and, when that line is executed, power will be sent to that output connection. A 0 means that no power will be sent. The two bits 7 and 6 are input bits. No power is sent to these bits, instead they are read to see if they are on or off, that is depending on the state of the opto-sensor brick. This information can be used to condition what action is performed next.

Values

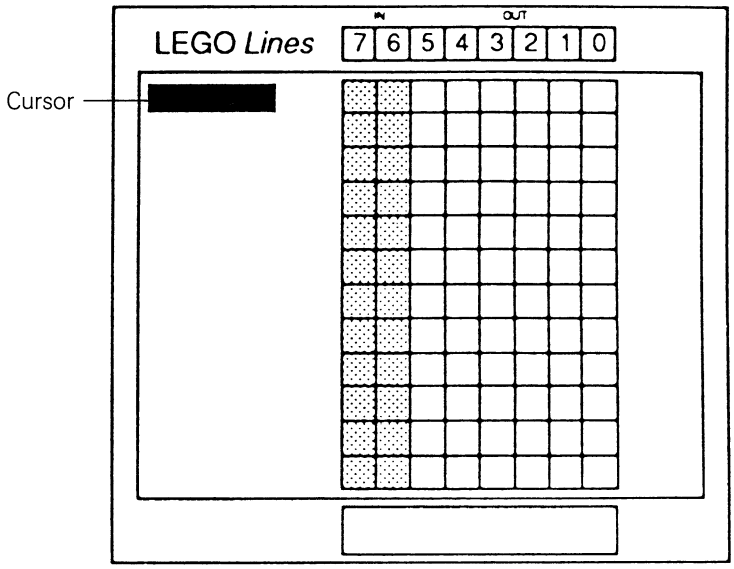
The value controls either how long an output activity is performed for, or how many times the activity will be executed before moving on to the next set of instructions.

Creating a *Lines* program

1 Getting started

The ***Lines*** desk is an autoboot disk, which means that it will load and run automatically. Put the ***Lines*** disk in drive 1, turn the computer on and then close the drive door.

There is a short pause before the main ***Lines*** program is loaded and the ***Lines*** editing screen appears.



The ***Lines*** editing screen

2 Entering information

In explaining the facilities and features of LEGO ***Lines*** the following keystroke convention is used:

`Apple-Key`	means hold down the ⌘ key, press the second key and then release the ⌘ key.

This section uses only keyboard facilities to operate the LEGO *Lines* program. The Mouse can also be used. Full details of Mouse operation, menus and special options can be found in the *Technical reference* section, page 4.7.

(a) Entering labels

The letter keys are used to enter labels.

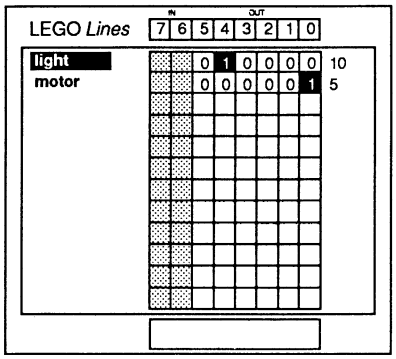
(b) Entering numerical or time values

The number keys 0–9 are used to enter numerical or time values.

(c) Setting a bit pattern

Holding the Apple key while pressing the number keys 0–7 will set the bits either off or on. The Apple-number key combination 'toggles' the state of that bit.

The middle section contains eight bits in a layout resembling that on the interface. The positions are marked, from left to right, 7, 6, 5, 4, 3, 2, 1, 0. The numbers 7 and 6 represent the input bit patterns and the others represent the output bit patterns (**R8, R9**).



Testing or running a line will mean that ***Lines*** will activate the interface to send power to all the output bits which contain a 1 — that are on — in that line. Input bit patterns have green indicator lights, output bit patterns have red indicator lights.

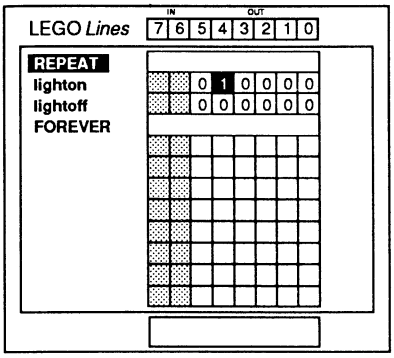
Initially, all the output bits are set off and the input bits have no state, described as **any value**, meaning that they are not to be tested at all in that line.

Bit patterns set the LEGO Interface and will only change when a new bit pattern is output. The bits will remain on until they are explicitly altered, either by another line or by the end of the program.

An output bit pattern can be entered on its own, that is to say that it does not require a label to be present. Input bit patterns can only be set, however, when the line starts with a keyword.

A note on keywords

A label is a descriptive word which is entered to identify the line by its function. In the same area, however, ***Lines*** will accept a number of command words, ***keywords***, which have special meaning in terms of function. They are used to allow loop structures (an activity or group of activities being performed repeatedly) and condition structures (activities being performed as a result of a decision to proceed) to be incorporated in the program design.



There are several keywords in LEGO ***Lines***. These are:

```
REPEAT
UNTIL
FOREVER
ENDREPEAT

IF
ENDIF

COUNT
```

These keywords are entered in the same way as the labels. The letters of the words are entered in lower case until the full keyword is typed, then the word is reprinted in upper case automatically.

See page 1.20 for an explanation of the use of these key words.

(d) Editing an instruction line

Alterations are made by moving the cursor (using the ↑ and ↓ arrows) to the line which requires changing. If a bit pattern needs alteration then the appropriate change can be made simply by switching the required bits off and on in the usual way. Changing a label or a value involves pressing the correct arrow keys and pressing **DELETE** to remove the characters.

The entire line may be 'blanked' out by pressing **Apple-B**. The entire line may be deleted by pressing **Apple-D**.

Pressing **Apple** ↓ takes the cursor to the last line of the program.

Pressing **Apple** ↑ takes the cursor to the first line of the program.

(e) Editing keywords

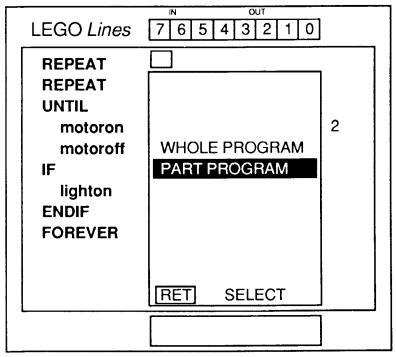
Keywords can be edited by pressing **Apple-B** or **Apple-D** to remove the whole keyword and retyping.

(f) Testing a bit pattern

Using the arrow keys to position the cursor on the desired output bit pattern and pressing **Apple-T** transmits that pattern to the LEGO Interface and will continue to do so while the keys are held down. This feature is useful in checking that the outputs are connected as required before actually running a program.

3(a) Running a whole program

Pressing **Apple-R** causes the program to be executed. The cursor block will move to the first line of the program and will then step through the lines as each is performed.

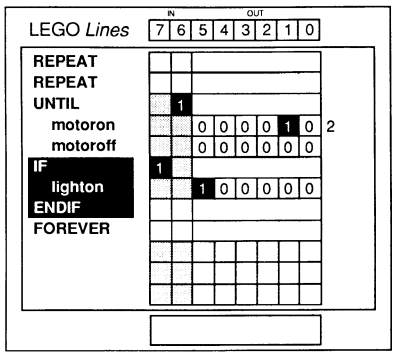


The **ESCAPE** key is used as a stop button if program execution is to be abandoned. Because the performance of a line may take a little time, the **ESCAPE** key needs to be held down until the program has definitely halted.

Pressing **SPACE** will halt program execution, pressing **SPACE** again continues the program.

3(b) Running part of a program

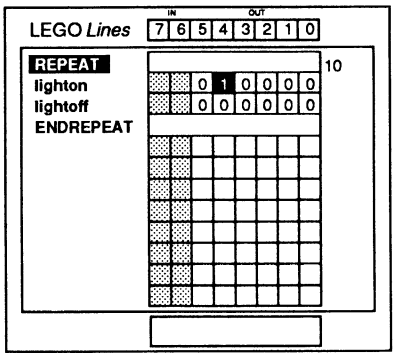
In addition to running a single line, ***Lines*** allows the testing of a group of lines. This can be done by positioning the cursor at the beginning of the part of the program to be tested and selecting **RUN** and **part program** from the menu (**R10**).



Use arrow key ↓ to highlight portion to be run and press **RETURN**.

Be careful that structures are complete within the portion to be tested.

Keywords and control structures



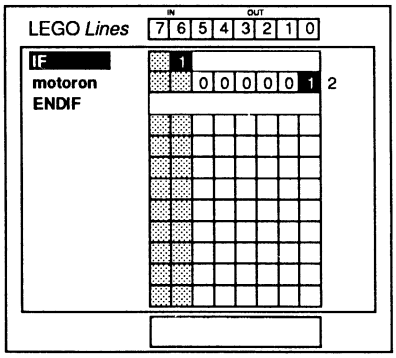
REPEAT FOREVER

When a set of lines is to be repeated indefinitely then the keyword **FOREVER** can be used. The lines ended by **FOREVER** will be repeated continuously or until the **ESCAPE** key is pressed.

REPEAT *n* ENDREPEAT

This allows the same lines to be repeated a set number of times. This number is entered in the value column. The structure is ended with an **ENDREPEAT** command.

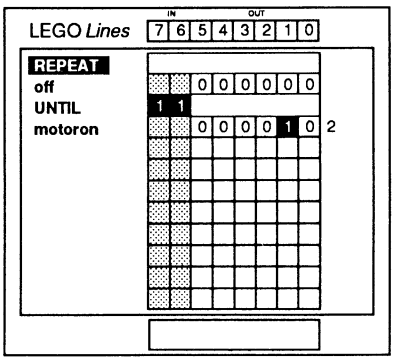
Condition structures



IF ENDIF

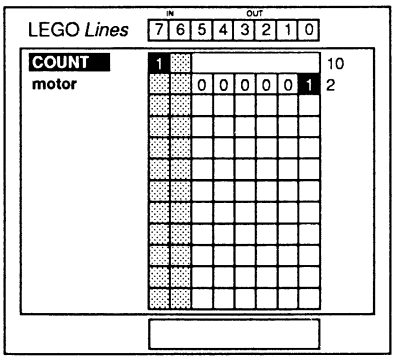
Lines can be skipped over or operated upon according to whether a certain bit pattern (the condition) is recognised. The **IF** structure is ended by **ENDIF**. The input bit pattern to be tested is set using the function keys **Apple-7** and **Apple-8**. If the bit pattern programmed into the input bit locations is the same as the bit pattern being received through the interface, the lines within the **IF ENDIF** structure are acted upon.

If the bit pattern being received through the interface is not the same, the next line to be executed will be the line following the **ENDIF**. There are two input bits and both may be used to set the input pattern to be tested. If only one input bit pattern is required then the other bit pattern may be cancelled by pressing **Apple-7** or **Apple-6** again until **any value** is shown.



REPEAT UNTIL

REPEAT structures can end with the **UNTIL** condition. This is sometimes also known as a **DO UNTIL** structure. This operates in exactly the same way as the **IF ENDIF** structure except that the input pattern is tested at the end of the lines included in the loop, rather than at the beginning. The lines will, therefore, be repeated until a certain input bit pattern is recognised and then the line after the **UNTIL** will be executed.



COUNT *n*

This keyword is used in conjunction with the LEGO black and white segmented disk. **COUNT** a number (*n*) of times can be used to count changes on the disk from black to white or from white to black. The **COUNT** function will only count on one of the input bit channels, either bit 7 or bit 6. The input bit not being counted will automatically be set to **any value**.

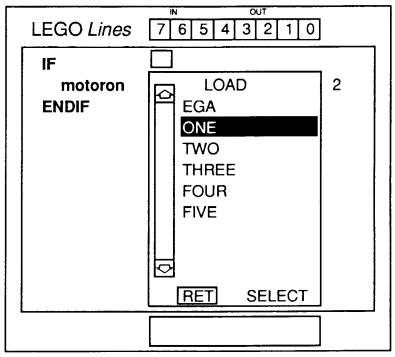
When **COUNT** is not the first line of a program the computer will continue to run the output line before **COUNT** until the computer has counted to the value specified. An example of this application can be seen in the **CONVEY** program, page 1.41.

Lines utilities

Lines includes a number of functions which enable the student to perform a number of other operations.

Loading and Saving *Lines* programs on disk

A program can be loaded or saved into ***Lines*** either by using the disk menu or pressing **Apple-L** for load or **Apple-S** for save.

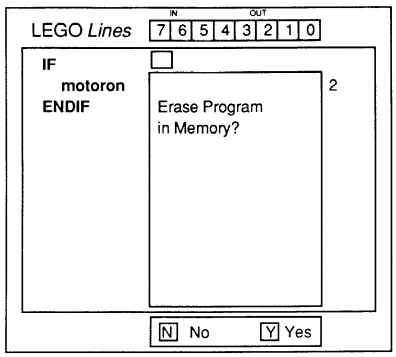


1 Loading

Press **Apple-L** and you will see a list of files in the disk. Select the file you want using the arrow keys to move the cursor. You can 'scroll' the file list if there are more files on the disk than can fit in the window.

When the cursor is where you want it press **RETURN**.

Note: If there is a program already on the screen you will be asked **Erase program in memory?**. Pressing **Y** will complete the instructions, pressing **N** will return you to the edit screen.



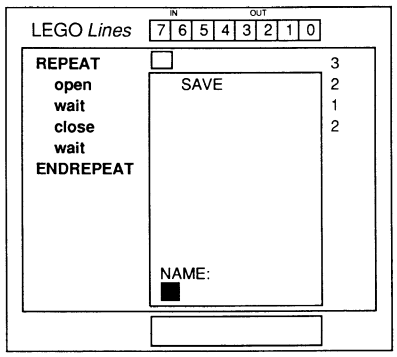
2 Saving

When saving a ***Lines*** program the following prompt will appear:

```
NAME
```

Type the name. The first character must be a letter. You can have up to ten characters for the name. You can correct typing errors using **DELETE**. Pressing **RETURN** will then save your program.

Note: If a file of that name already exists on the disk you will be asked, **Name exists. Replace?**. Press **Y** to replace, press **N** to return to edit screen.



3 Display the contents of the disk

During load and save operations, it is often useful to be able to read what is on the disk, to confirm that a program has saved correctly or to list the programs available to load. This option is available from the disk menu, or by pressing **Apple-L**.

4 New program

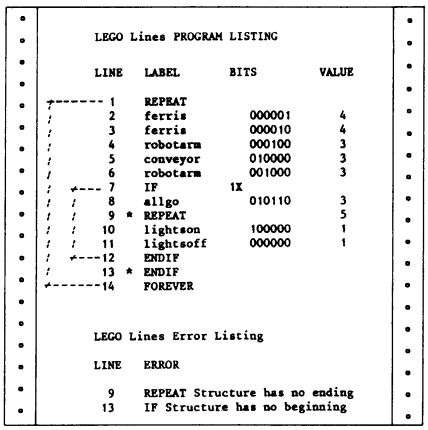
An entire program can be deleted from memory by selecting **NEW** from the MISC menu. Before the program is replaced you will be asked **Erase program in memory?**.

The current program is removed and blank ***Lines*** editing screen reappears. A program which has been wiped cannot be retrieved unless, of course, it has previously been saved to disk (**R10**).

5 Printing a *Lines* program

Pressing **Apple-P** enables a printout to be made of the program you are currently working on. Refer to ***Technical reference*** section, page 4.11. You will be given a choice between printing to the screen or to a printer. The printout will contain the labels, bit patterns and values together with an outline of their structures and a listing of errors, if any. The error lines will be marked with an asterisk.

Note that trying to print without a printer connected will cause your program to be lost unless you have already saved it to disk. Pressing **ESCAPE** during printing will return you to edit screen.



```
LEGO Lines PROGRAM LISTING

      LINE   LABEL       BITS          VALUE

 ┌------ 1   REPEAT
 ¦       2   ferris        000001        4
 ¦       3   ferris        000010        4
 ¦       4   robotarm      000100        3
 ¦       5   conveyor      010000        3
 ¦       6   robotarm      001000        3
 ¦  ┌--- 7   IF                1X
 ¦  ¦    8   allgo           010110        3
 ¦  ¦    9 * REPEAT                        5
 ¦  ¦   10   lightson        100000        1
 ¦  ¦   11   lightsoff       000000        1
 ¦  └---12   ENDIF
 ¦      13 * ENDIF
 └------14   FOREVER

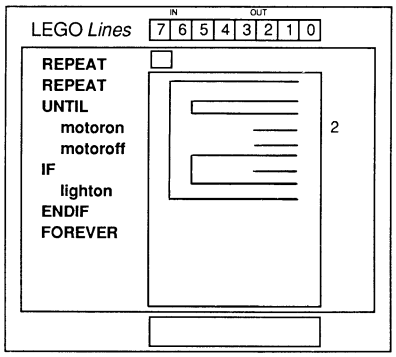

LEGO Lines Error Listing

LINE   ERROR

   9   REPEAT Structure has no ending
  13   IF Structure has no beginning
```

6 Viewing the structure

This feature allows you to view a structure of the program to see how it behaves or to notice some errors. **Apple-V** will display the structure as a series of lines in the middle of the screen. **IF** and **REPEAT** structures will be represented as loops. Errors will also be displayed.



7 Using ESCAPE

The **ESCAPE** key can be used as a means of cancelling the operation.

8 Ending a *Lines* session

Selecting **EXIT** from the Apple menu is the means by which you can end a ***Lines*** session. A warning will appear asking whether you really wish to proceed. If you confirm that you wish to exit, the screen will be cleared and ***Lines*** will be cleared from memory.

9 Other utilities

Other utilities which include sound on/sound off facilities, help with key functions, erasing programs from the disk, etc, are described in the ***Technical reference*** section, page 4.11.

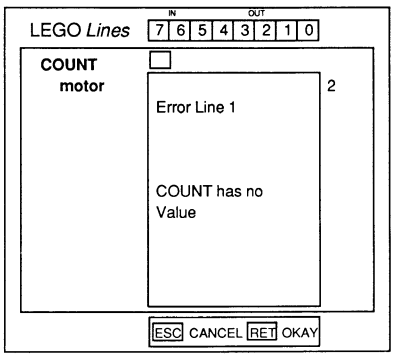
Error messages

Before running or viewing a program, ***Lines*** does various checks to ensure the structures within the program are correct. If there are any errors then a message will be displayed in the form:

ERROR LINE (line number)

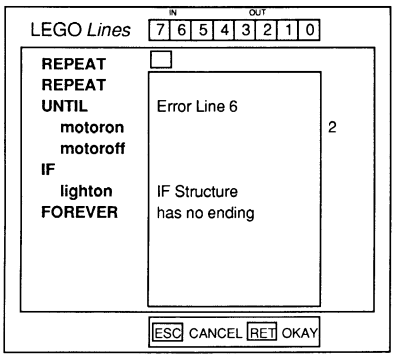
Below this message the error messages appear as follows:

1 **Structures too deeply nested**
2 **COUNT has no value**
3 **If structure has no ending**
4 **Repeat structure has no ending**
5 **If structure has no beginning**
6 **Repeat structure has no beginning**



Explanations of error messages

1 You have more than eight levels of nexting in the program structure.
2 You need a numerical value in the line containing **COUNT**.
3 Either there is an **ENDIF** missing or the **IF-ENDIF** structure is broken.
4 Either there is an **ENDREPEAT/UNTIL/FOREVER** missing ***or*** **REPEAT** has a wrong ending, ***or*** the **REPEAT** structure has been broken.
5 **ENDIF** without **IF** or the **IF** structure is broken.
6 **ENDREPEAT**, **UNTIL**, **FOREVER** have no **REPEAT**, ***or*** start the wrong way ***or*** the structure has been broken.



After each message is displayed you may press **RETURN** to continue the check. Pressing **ESCAPE** will return you to the edit screen.

A program with errors will not run!

Disk related errors

The possible disk errors are:

Disk full
Bad disk (includes unformatted disk)
Disk Write-Protected

LEGO *Lines*: program design

This section is designed to assist the teacher and, consequently, the student in the design and testing of programs using the LEGO ***Lines*** software environment.

Suggestions for the organisation of pupil-based work are also included, reflecting the overall philosophy of the package and the fact that pupils should become capable of solving a range of technological problems using the materials provided.

Three aspects of program design are discussed:

— designing algorithms and testing them using the software
— identifying key software techniques from worked examples
— suggestions about the many tasks pupils could perform using this package.

Algorithms

An algorithm is a set sequence of steps which describes a solution to a particular problem.

There are a number of ways to go about designing a computer program but here is a method which the authors recommend:

1 Analyse the problem by producing a logical statement or a sentence of what the solution(s) to that problem may be. A complex problem will have to be broken down into a number of smaller sub-problems; each of which should also be described in sentence form.

2 Identify the logical structures which are present in these sentences. The use of a ***structured flow diagram*** is usually of great assistance, especially as this can be compared with the boxed display, available as a software utility.

3 Identify the program instructions which correspond to the structures previously identified.

4 Link the sub-problems together to form a complete solution to the problem.

5 Test the solution(s) and make appropriate changes to the program if necessary.

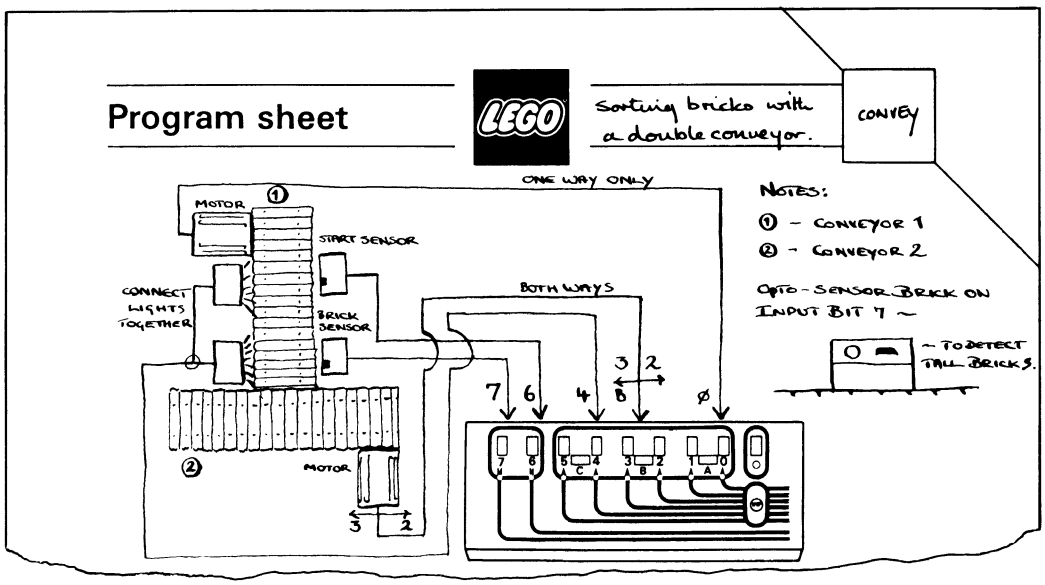
6 Choose the most effective solution.

The development of a computer program is an example of the larger technological problem-solving process at work and it should be subjected to the same rigorous testing as all technological activities.

As you progress through these steps, it will become necessary to convert statements like 'switch motor on' to lines of instruction, involving the creation of bit patterns. The following has proved a useful guide for this process:

1 Define the use of the interface by drawing a schematic diagram of the model being used and identify which bits correspond to which input or output connections. (The program sheet has been designed to facilitate this.) An example is shown below.

Example: Double conveyor used to sort large and small bricks

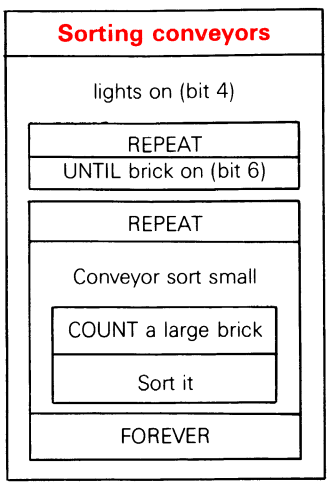


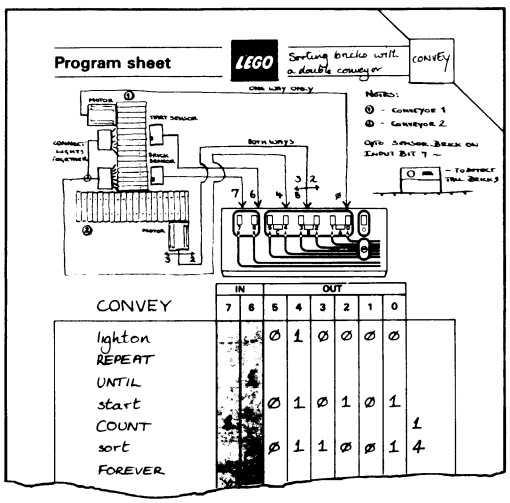
2 For each of the sentences or blocks in the structured flow diagram identify:

- the starting values of the bits
- the changes in those bit patterns once they are subjected to the processes of the algorithm
- the end values of the bit (that is, the start values of the next sentence or block).

3 Mentally trace the algorithm through line by line focussing on the bit pattern as it changes.

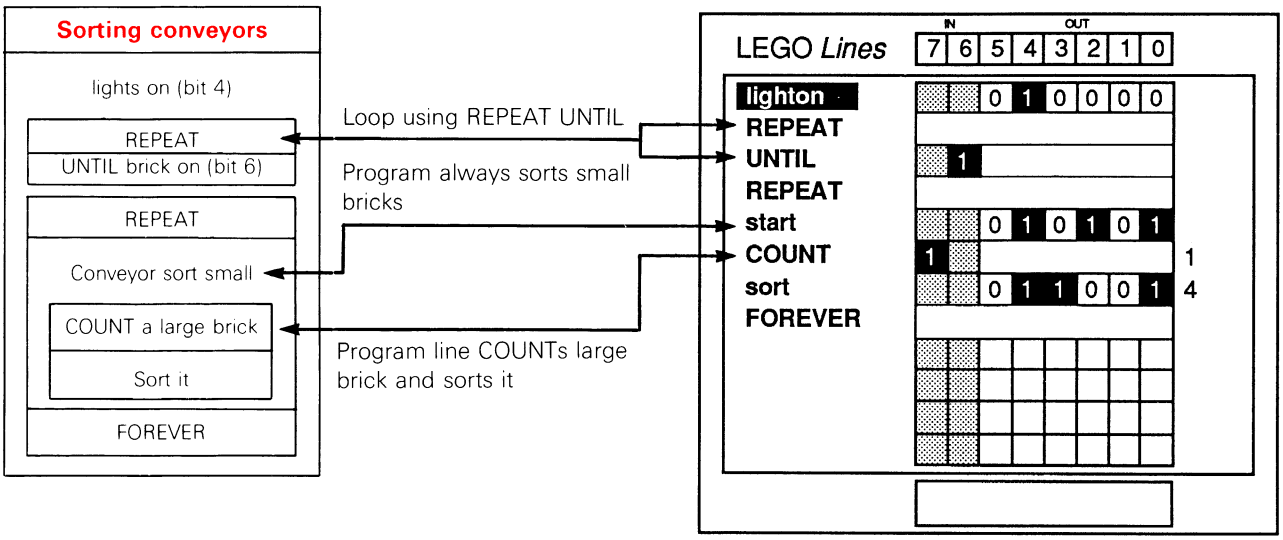
4 Carefully observe that the model behaves as you expect it to.





Key software techniques

Twelve software functions are introduced in the students' assignments and these are detailed in the ***sample program*** section (page 1.31). Together with a **COUNT**ing function they form the essential introductory building blocks for programming.



The software environment allows the creation of programs of 30 lines maximum and containing up to 8 nested loop structures.

It is a very powerful aid to programming in itself, but it does have a limit. If two input sensors are used it is complex to construct a condition (**UNTIL** or **IF**) which will operate on one input **OR** the other at the same time.

Care must be taken when using **REPEAT UNTIL** structures where the **UNTIL** condition is waiting for an event to take place and there are other actions in the structure. Occasionally an input trigger or signal will appear to be missed. This is because the computer may not actually be looking at the **UNTIL** line at the time the opto-sensor is triggered. Good program design and appropriate use of light bricks (to force the required signal from the opto-sensor brick) will overcome this.

Some program builders' hints

1 Using REPEAT UNTIL structure

When you use **REPEAT UNTIL**, the functioning line executed ***before*** **REPEAT** will run continuously. The ***next*** line will not be executed **UNTIL** the opto-sensor is triggered (as in the example screen above).

2 Using REPEAT/ENDREPEAT and IF/ENDIF structures

When you use these structures the last line executed ***before*** the structure will run continuously until:

— you reach the end of the program
— you finish a count command
— it is replaced by another output line.

3 Using COUNT

When you use **COUNT** the last line executed before **COUNT** will run continuously until the opto-sensor has counted the desired number of times (only once in the sample screen above).

4 Inaccuracies in model positioning when using the COUNTing disk

When running programs with models incorporating the counting disk and the opto-sensor there may appear to be errors in model positioning (that is, returning robot arm to 'home', etc).

These errors will be due to the way the opto-sensor works and the physical size of the segments of the counting disk. Some errors are unavoidable in both the real world and in the LEGO model.

However, careful observation, elegant program design and the strategic use of sensors in the model environment will minimise or remove most positional errors.

Structured flow diagrams

The technique of employing structured flow diagrams is an important aid to program design. By using such a device students are forced to adopt a 'top-down' approach to solving a problem using the programming environment.

Example

Turn the washing machine and the wash cycle indicator light on when the door is closed and the switch is on. Turn the motor off when the door is opened, and the light off when the switch is off.

Identify structures:

Actions
Turn motor and light on
Turn motor off
Turn light off

Test
When door is closed and switch is on
When door is opened
When switch is off

Loops
Both off until door and switch on
Motor on until door opened
Light on until switch off

sequence ↓

Produce structured flow diagram

Washing Machine

REPEAT
UNTIL ON and shut

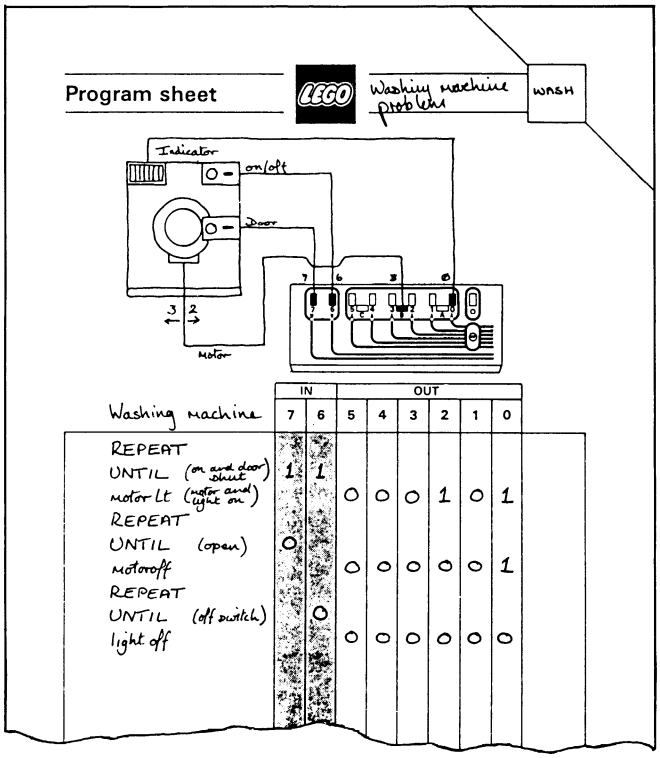
Motor and light on

REPEAT
UNTIL OPEN

Motor off

REPEAT
UNTIL switch OFF

Light off



All of the models will support many different programs some of which will perform a particular task better than others. A 'good' solution is one which does the job it is required to do and is elegant in design, in the sense that it is efficient yet uncluttered and easy to follow.

The following are samples of problems which can be solved using the washing machine model (1090 C) and the programming environment.

1 Turn the washing machine motor on and then off after a certain time has elapsed.

2 Turn the washing machine motor on for a certain time and indicate that it is on by a warning light. Then turn the light and motor off.

3 Turn the motor on for a certain time only if an on/off switch is turned on.

4 Turn the motor and indicator light on for a certain time only if an on/off switch is turned on.

5 Turn the motor on for a certain time only if an on/off switch is turned on and the door is closed.

6 Turn the light on when the on/off switch is on and then turn the motor on when the door is closed. The motor should not turn on unless the on/off switch is on.

7 Turn the washing machine on when the door is closed and the switch is on. Execute a wash cycle (drum turns one way for a short while, then the other) and a spin cycle (drum spins quickly, in one direction only) indicating when each is taking place.

8 Turn the washing machine on when the door is closed and the switch is on. Execute a wash cycle and a spin cycle indicating when each is taking place.

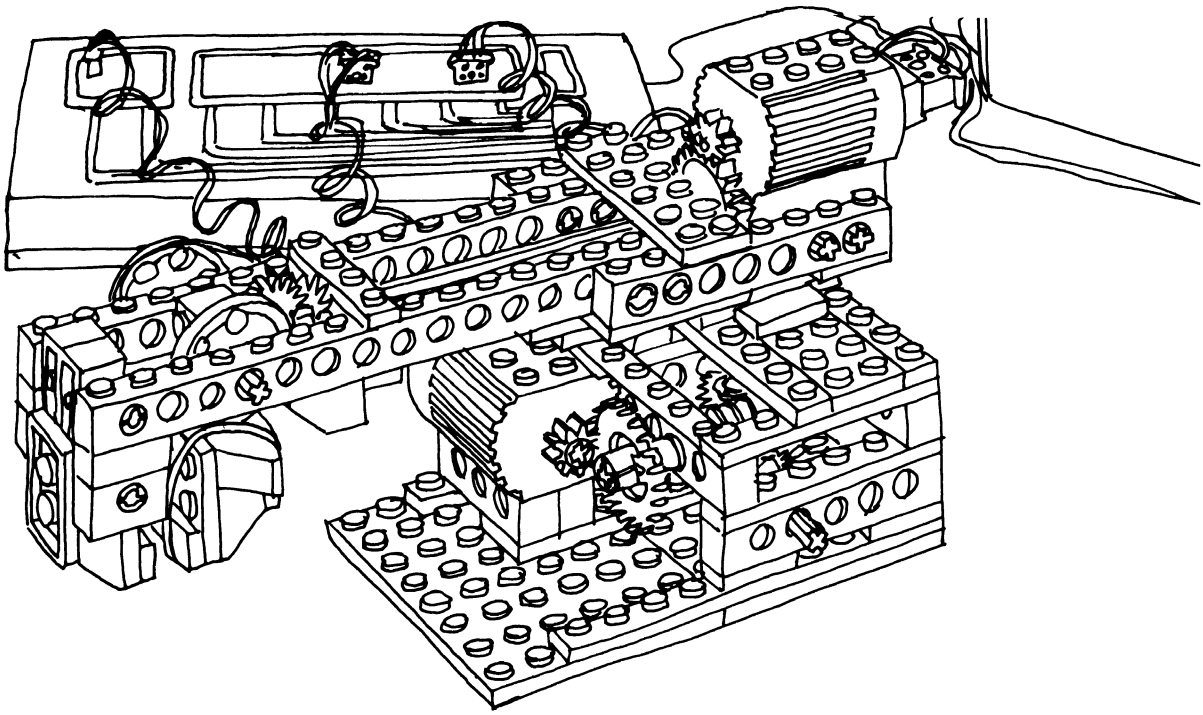
9 Incorporate an emergency switch into the programs.

Adapting or combining models

Pre-built models can provide a useful starting point but students will naturally suggest improvements to the design of particular models and ways of improving performance. It is possible, for example, to add a gearing mechanism to the washing machine so that the wash and spin cycles operate at different speeds. The Robot Arm model (1090 E) could easily be adapted to operate in an up/down plane, and so on.

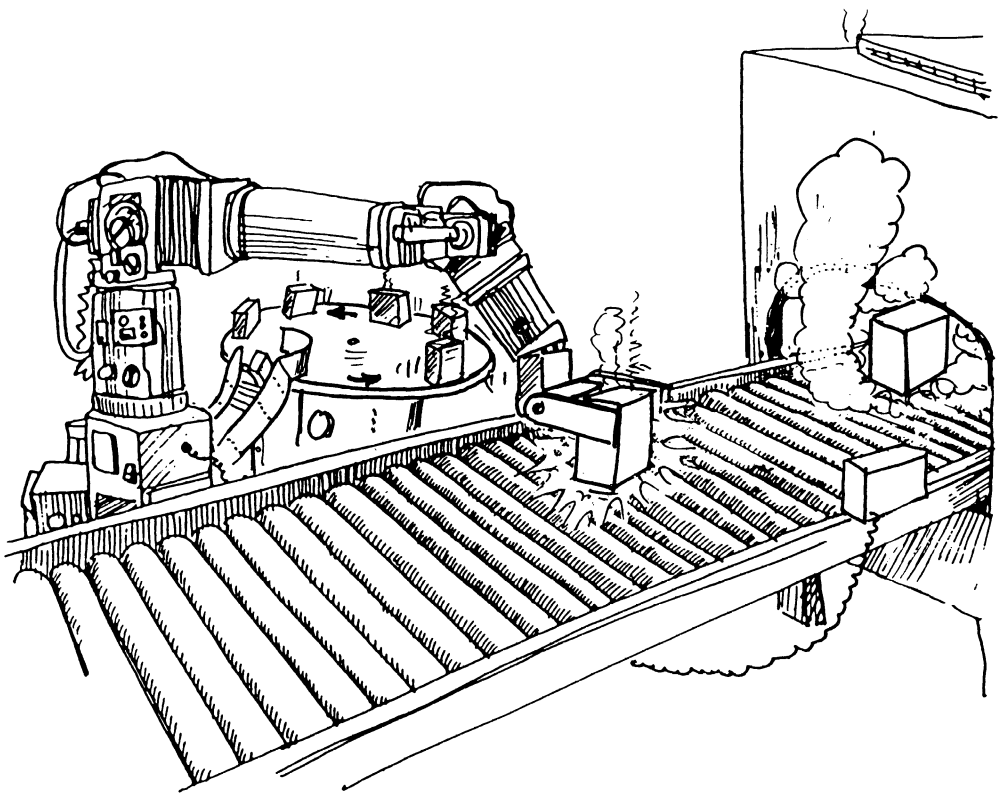
These extensions are to be encouraged as they promote the understanding that both software and hardware solutions combine to form the best solution to a particular problem and students need to understand this clearly. Selection of the most appropriate solution is a very important part of the technological process.

It is expected that as students gain in confidence they will wish to move away from the assignments into problems which they define for themselves. Indeed motivation for such activities will be high and results are likely to be good.

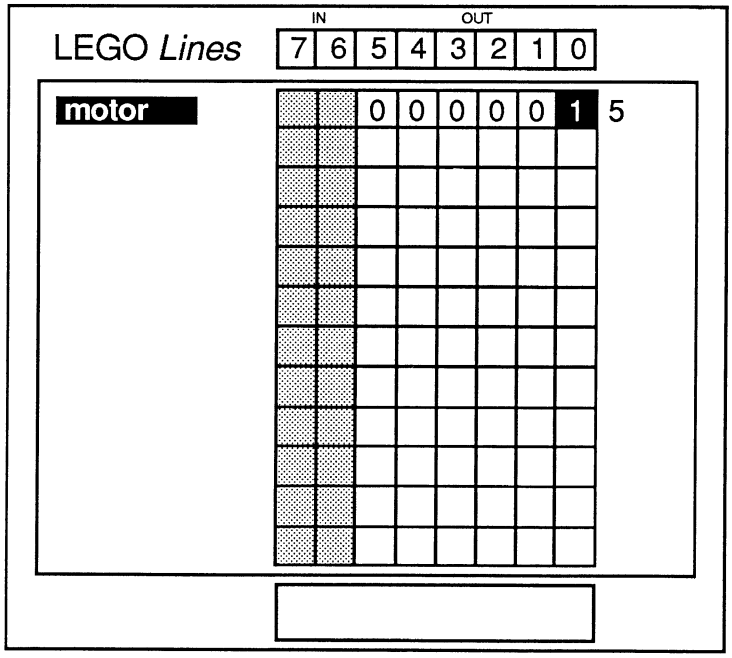


Open-ended problem-solving should be encouraged by discussions of particular environmental, manufacturing, or social needs. It will be important to consider the ways in which computer control has something more to offer than simple mechanical solutions.

It will also be important to refer the students to the problem-solving process they will have already experienced through the recommended course. This will now be a valuable resource.



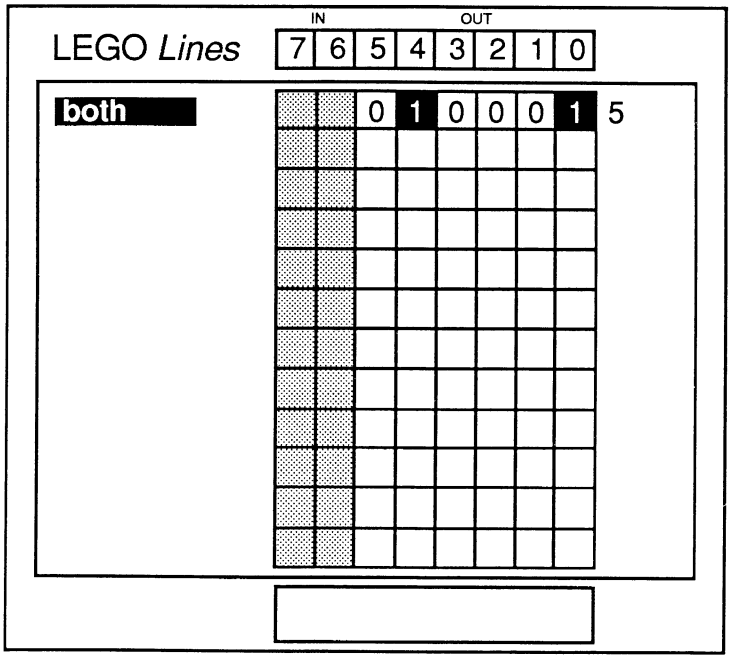
ONE



ONE
motor

This program demonstrates control of a single motor in one direction for 5 seconds.

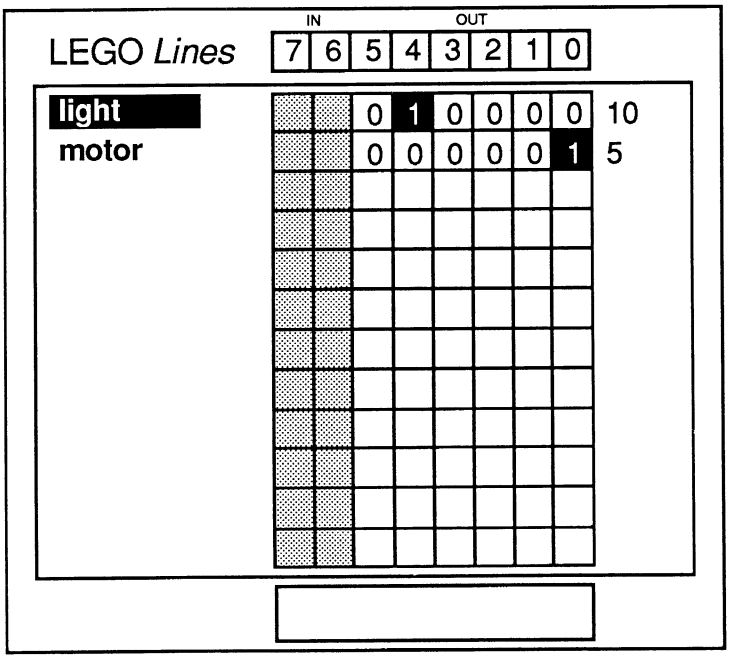
TWO



TWO
both

This program turns on a motor and a lamp for 5 seconds.

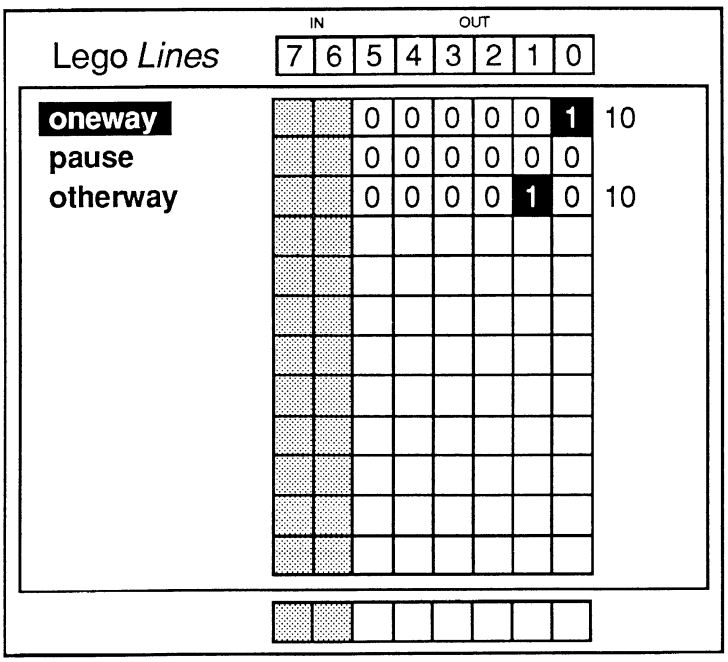
THREE



THREE
light motor

This program is made up of a sequence of two instructions.

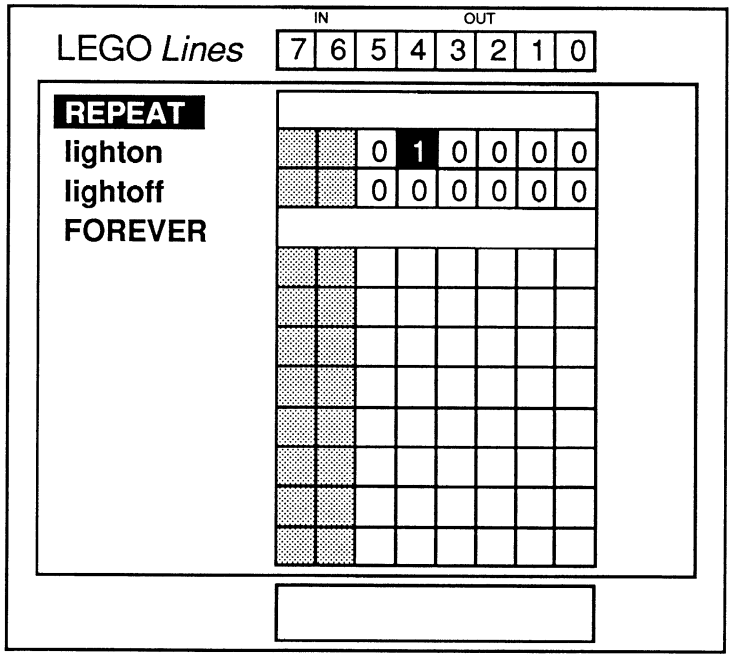
FOUR

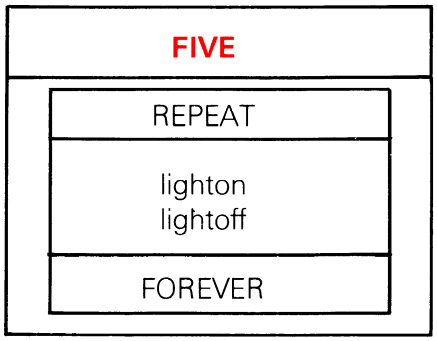


FOUR
oneway pause otherway

This program contains a sequence of three instructions which demonstrates rotation of a motor in clockwise and anticlockwise directions.

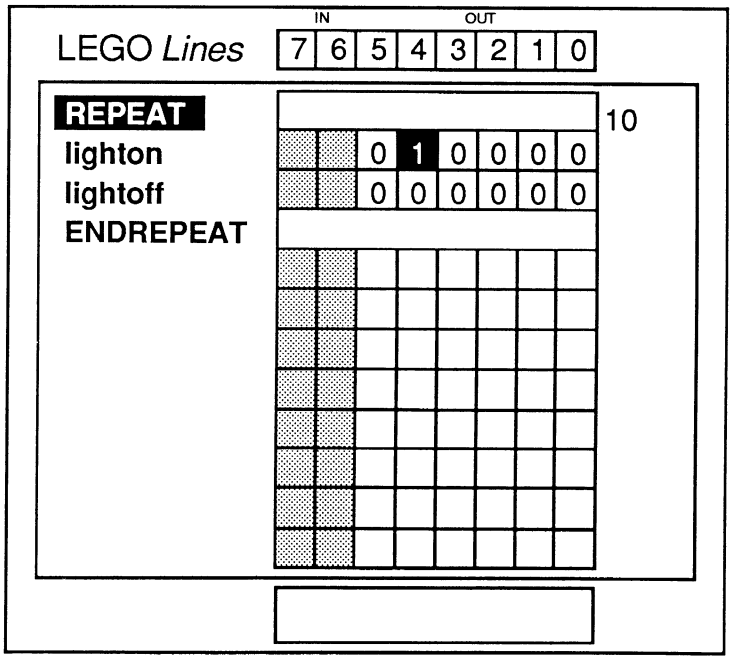
FIVE

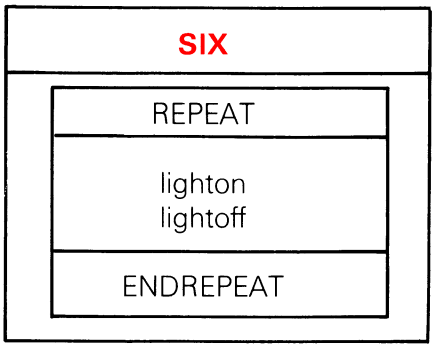




This program turns the lamp on and off forever (until **ESCAPE** is pressed).

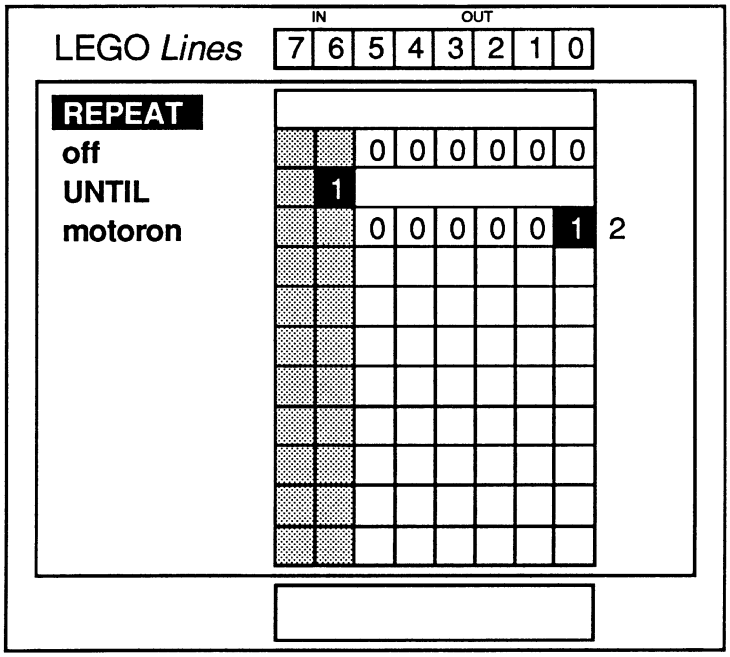
SIX

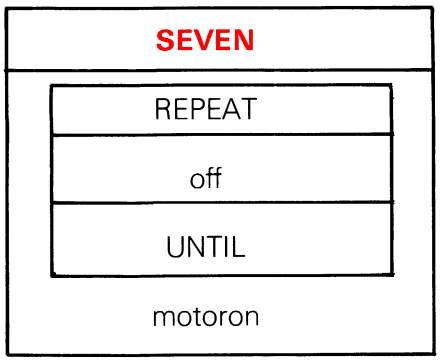




This program demonstrates a sequence of instructions being repeated a set number of times.

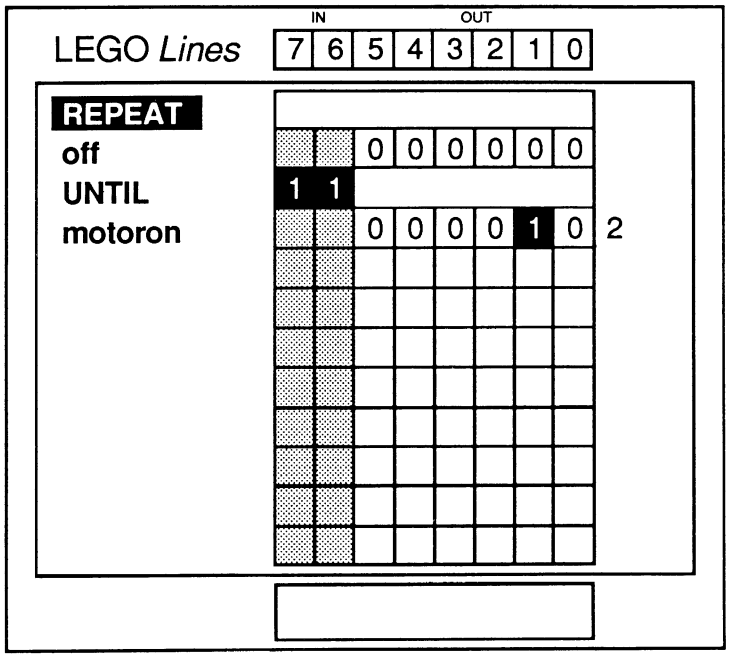
SEVEN

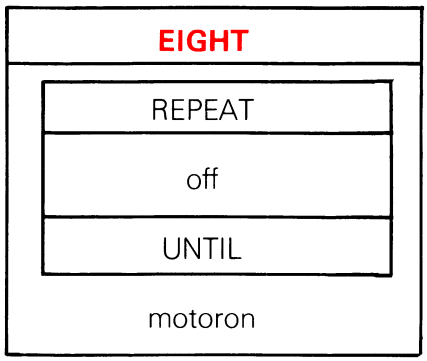




This program illustrates the use of **REPEAT UNTIL** using one sensor as an input.

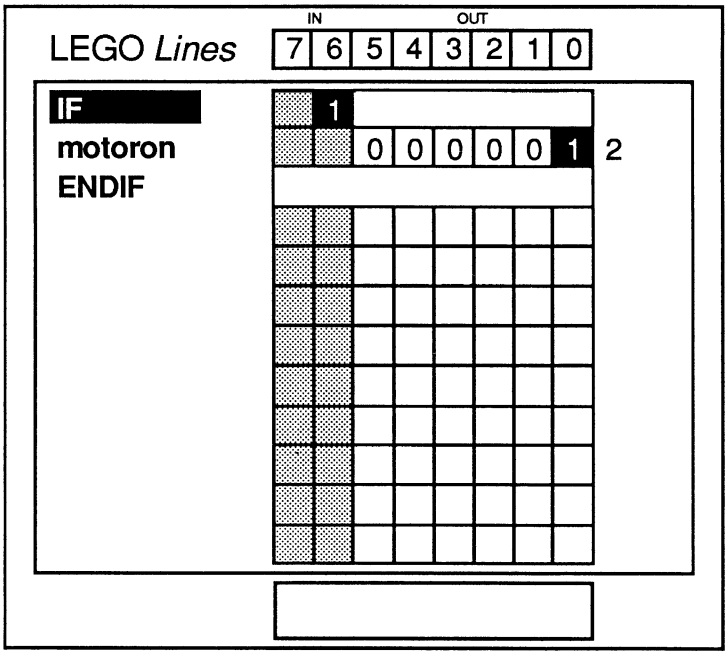
EIGHT

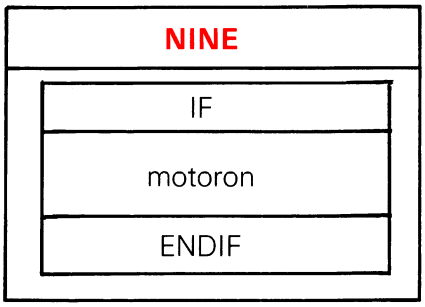




This program illustrates the use of two sensors and the **REPEAT UNTIL** structure.

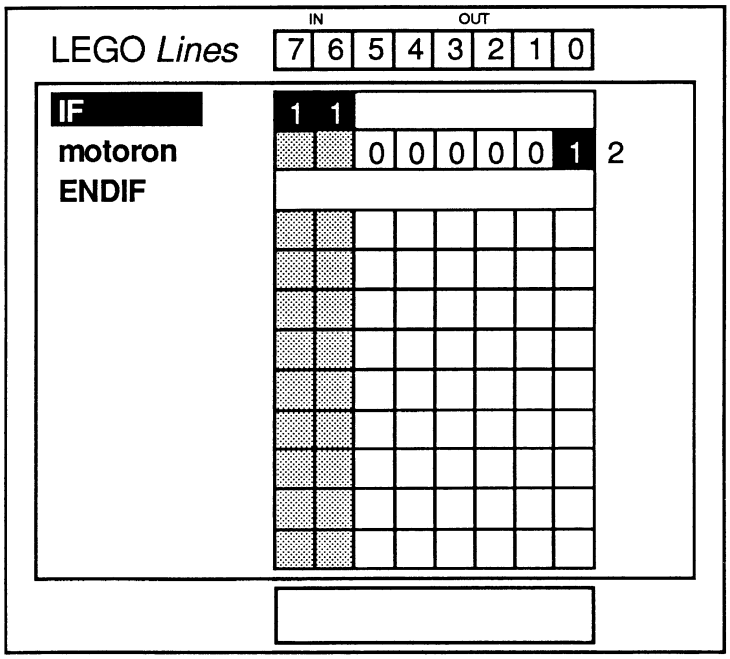
NINE

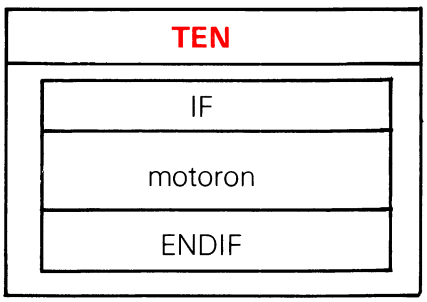




This program illustrates the use of **IF ENDIF**.

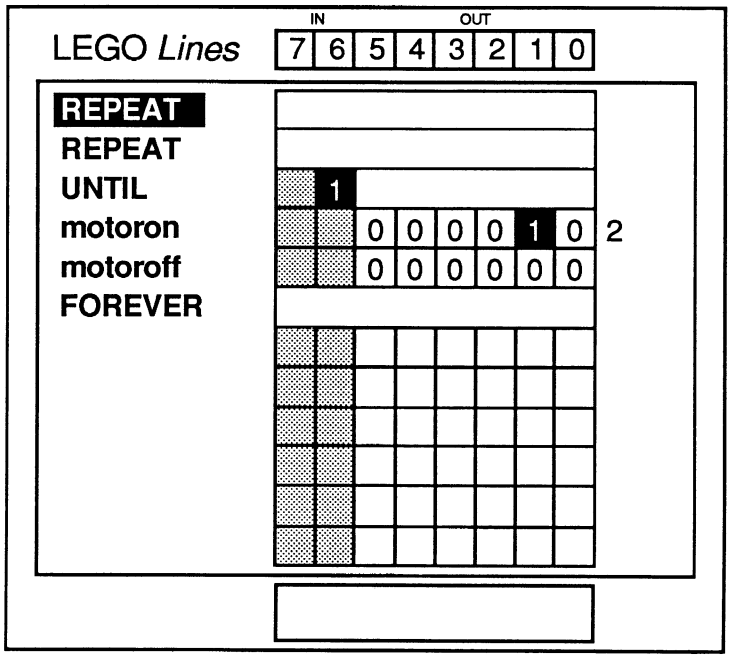
TEN

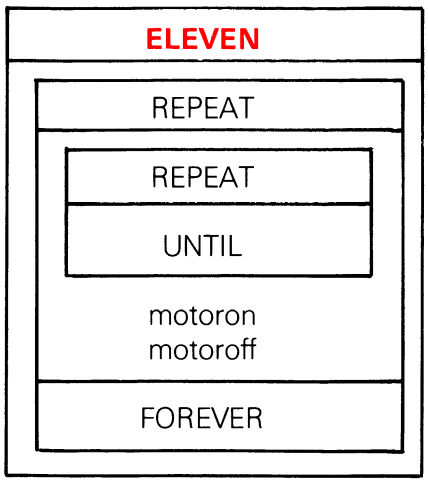




This program turns a motor on if input bit 7 and input bit 6 are both on.

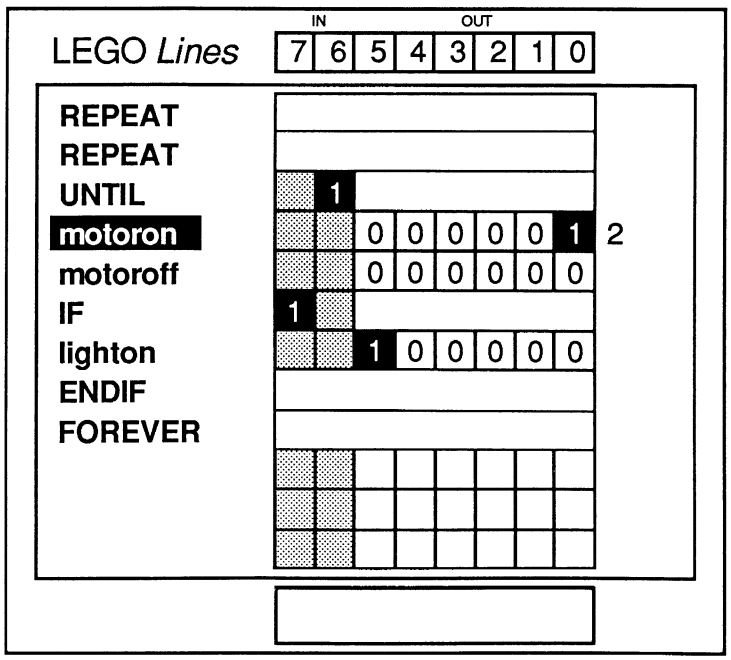
ELEVEN

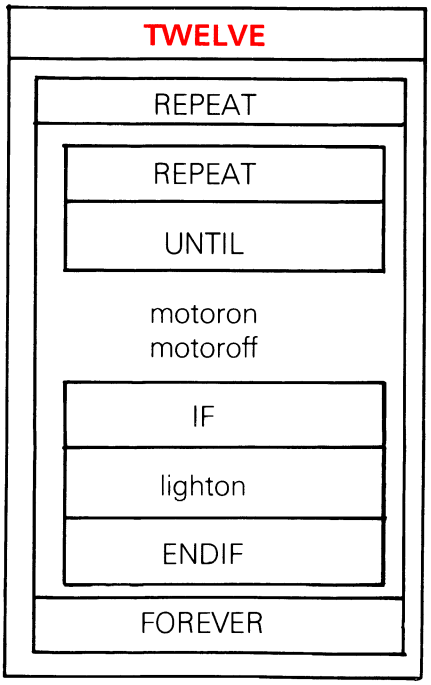




This program continually looks for an input and, when it has been found, executes a sequence of instructions.

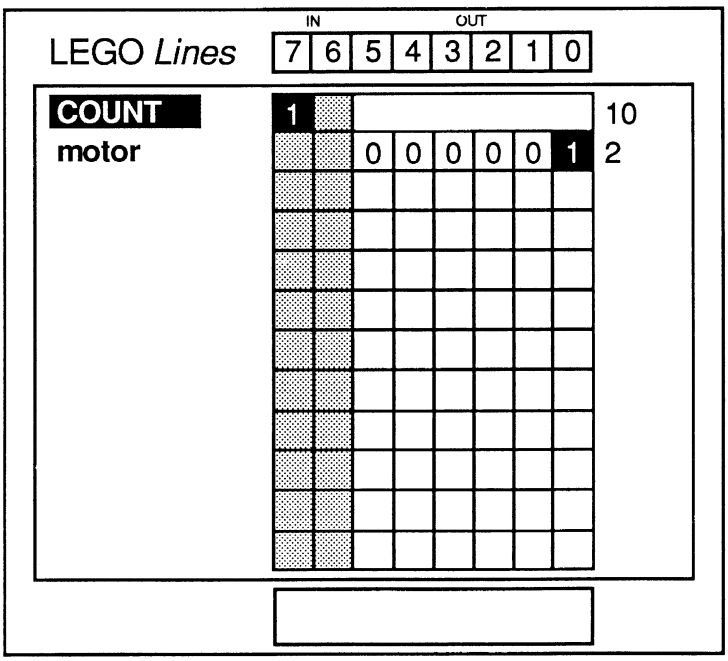
TWELVE





This program continually executes a sequence of instructions which includes an **IF ENDIF** structure.

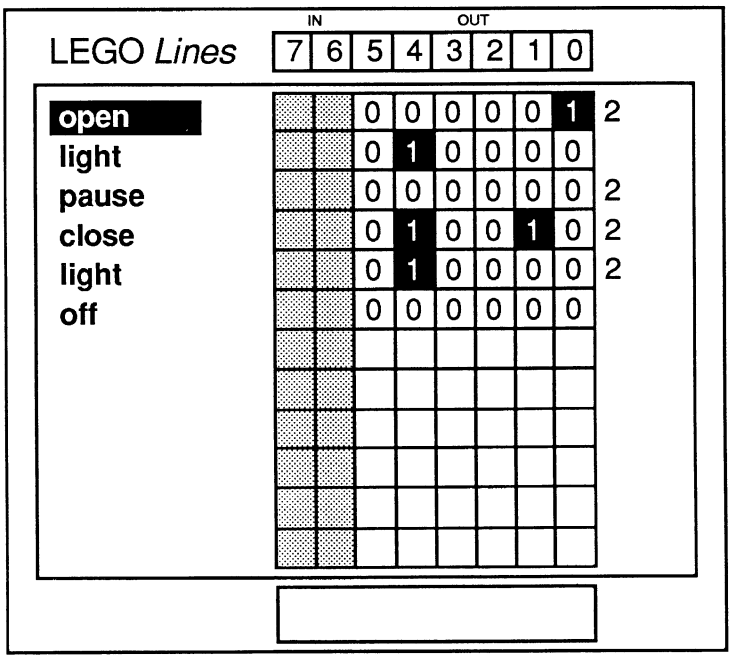
COUNT



COUNT
COUNT
motor

This program is designed to wait until the sensor is 'switched' 10 times before turning a motor on.

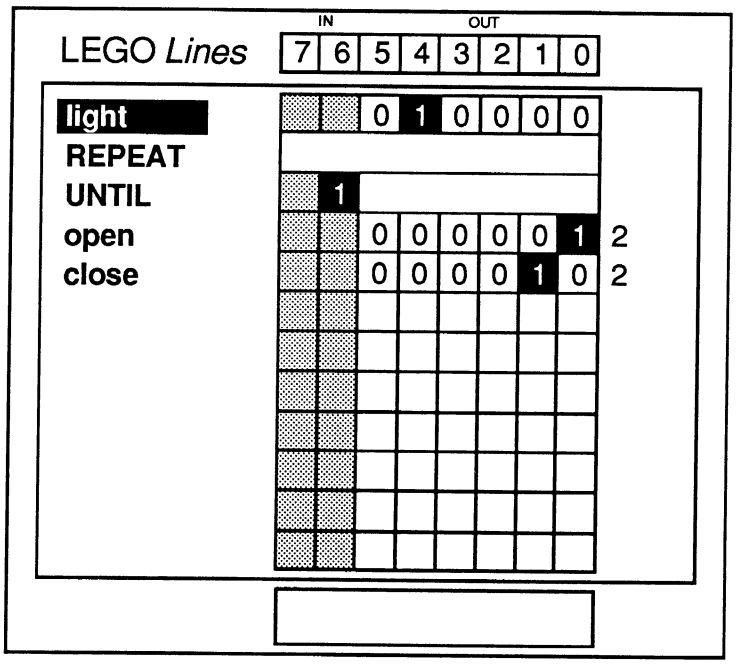
ATEST

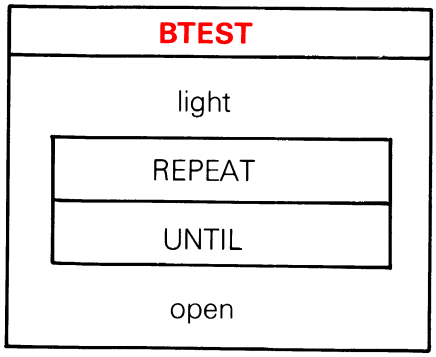


ATEST
open
light
pause
close
light
off

This program is designed to open the door and light a lamp. The door should pause before it closes and then the light is switched off after 2 seconds.

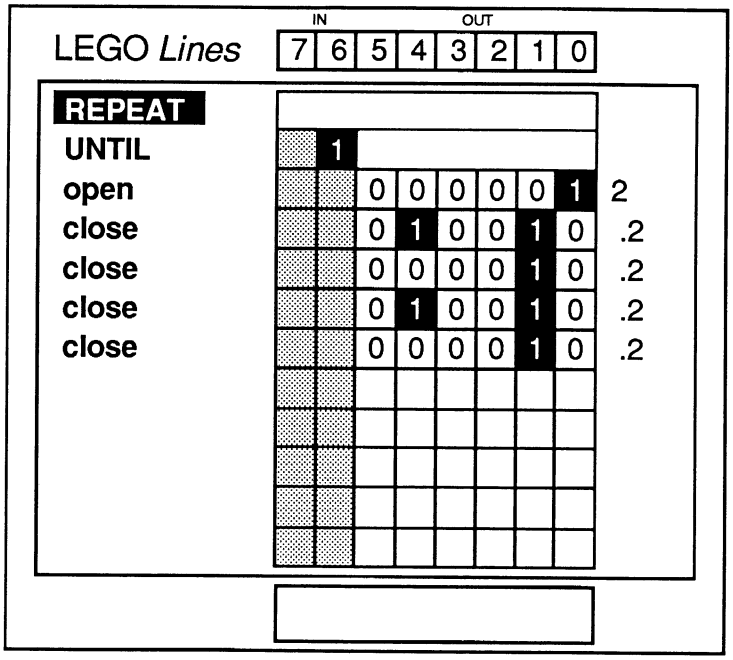
BTEST

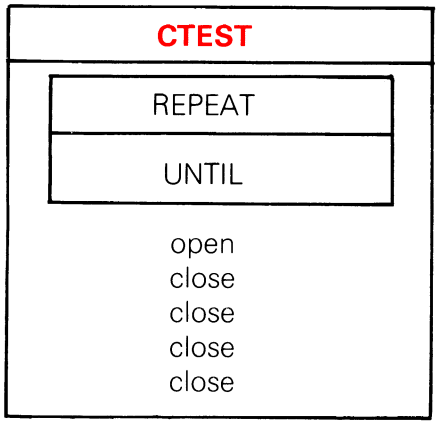




This program waits until input bit 6 is on before it opens and closes the automatic door.

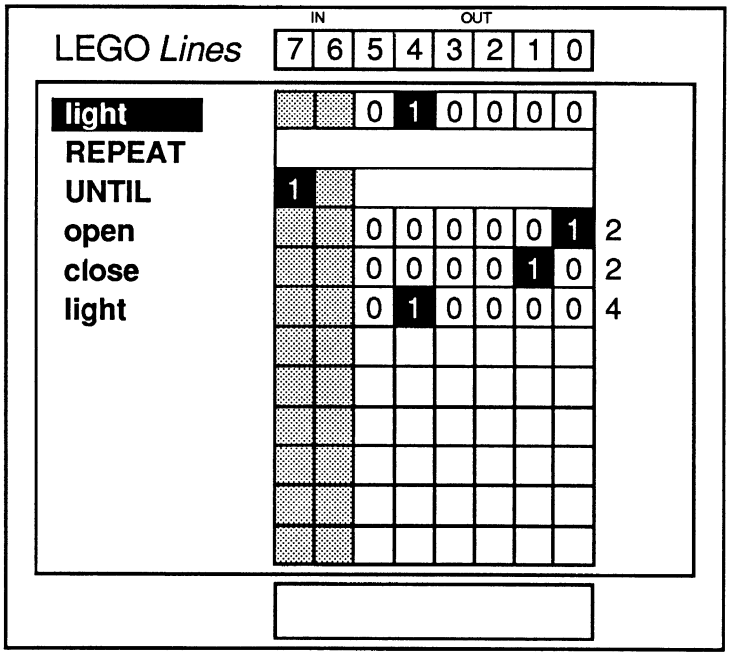
CTEST

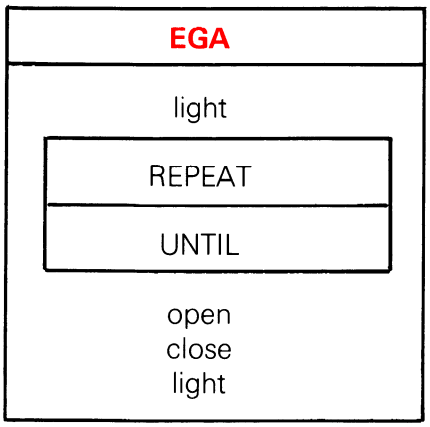




This program is designed to wait until an appropriate signal opens the door and then the light will flash on and off as the door closes.

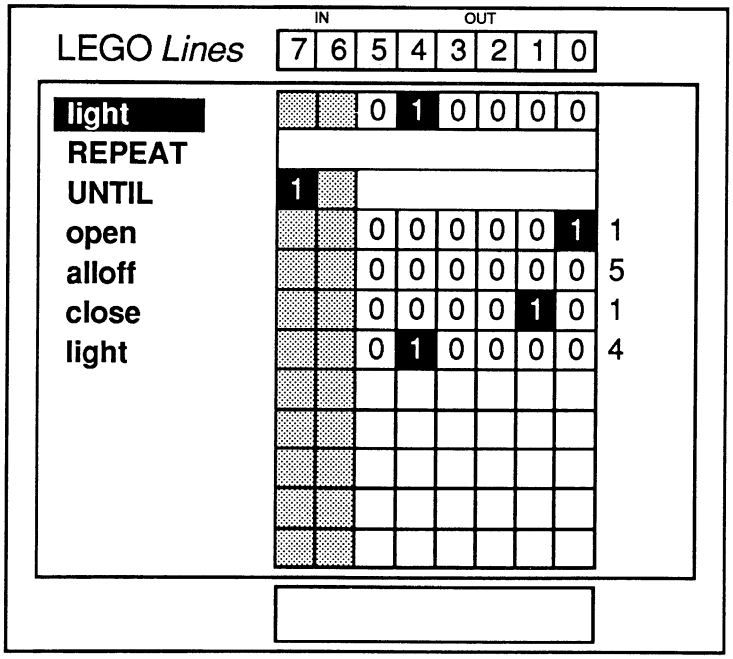
EGA

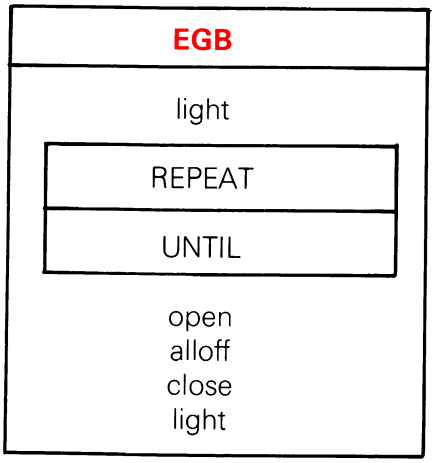




This program lights a lamp and waits until input bit 7 is made to switch the door to open and close. The light will stay on for 4 seconds when the door is closed.

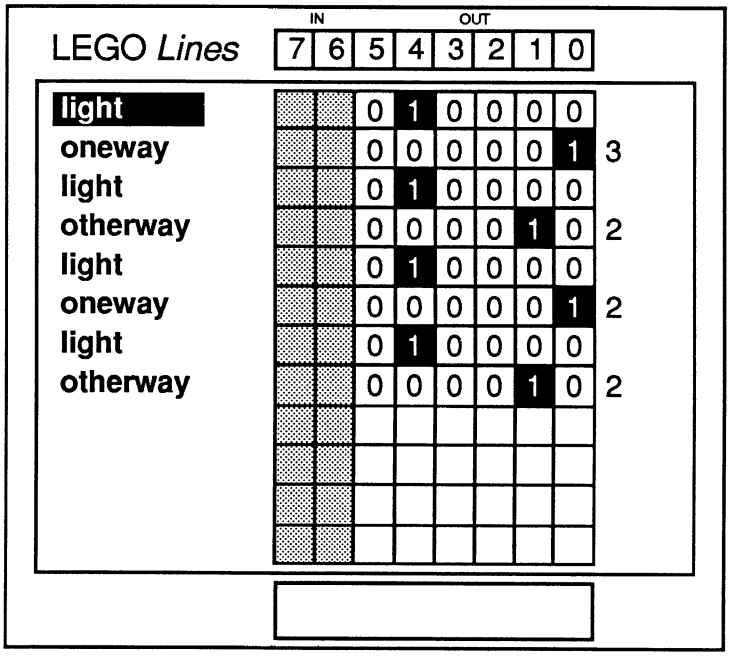
EGB





This program performs in the same way as the last program, except that it includes a new line, labelled *alloff*.

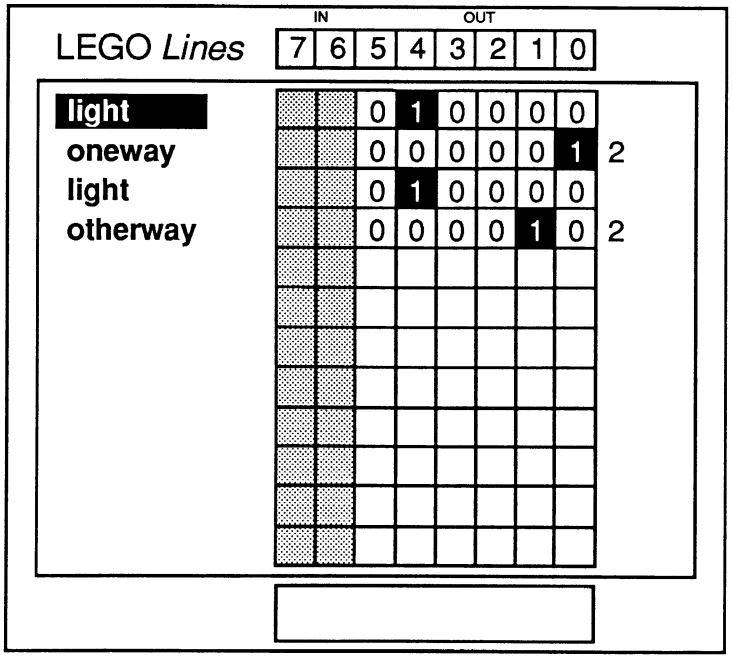
EGC



EGC
light oneway light otherway light oneway light otherway

This program demonstrates the use of output bits A to control a motor and output bit 4 to control a light. It is an extension of **MLA**.

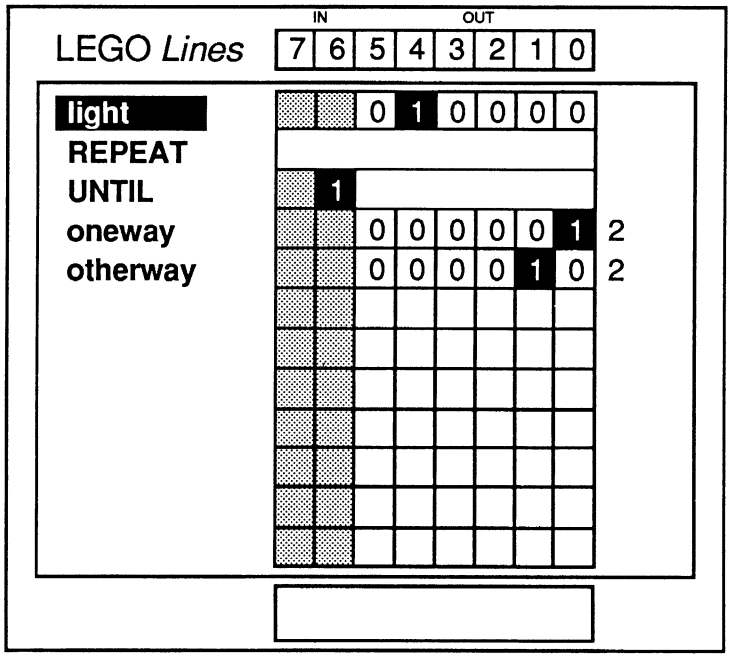
MLA

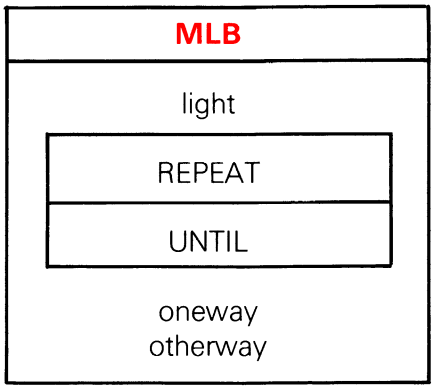


MLA
light oneway light otherway

This program demonstrates the use of output bits A (0 and 1) and 4 to control a motor and a lamp.

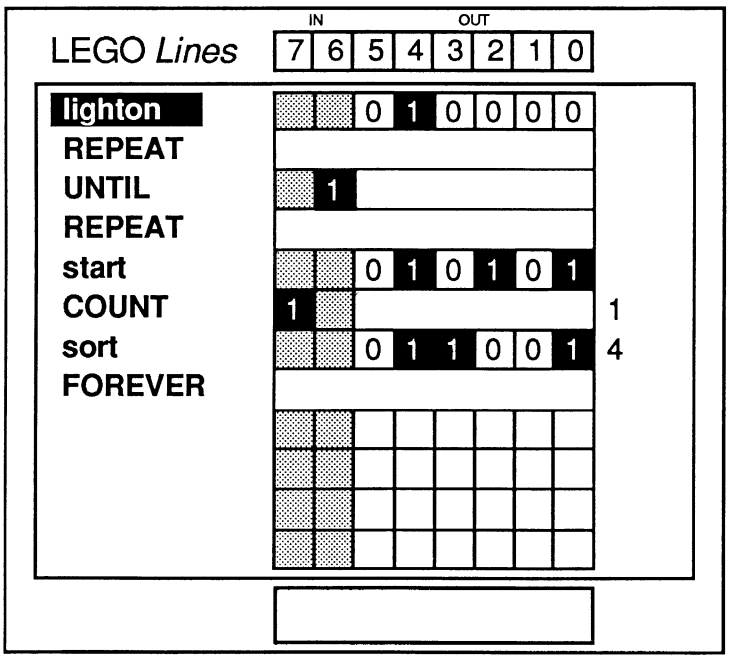
MLB

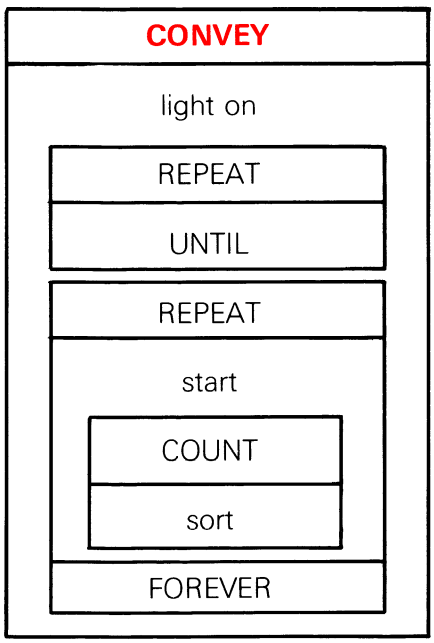




This program is designed to wait until a signal is received on bit 6 before the motor is switched on. The motor will rotate clockwise and anticlockwise for 2 seconds.

CONVEY





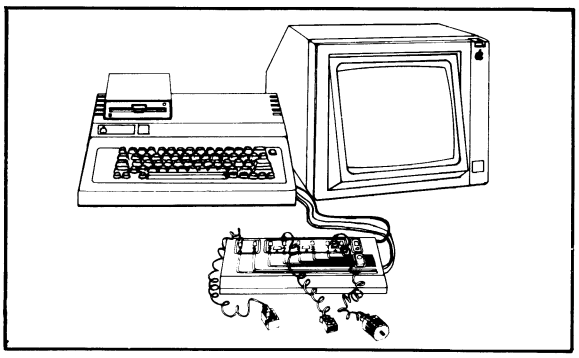
This program is designed to sort large bricks from small bricks on the double conveyor model.

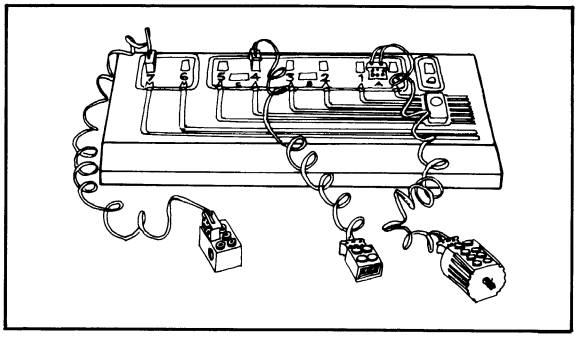
Section 2

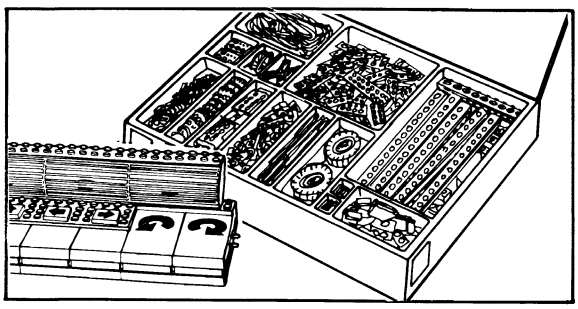
Classroom materials

Programmable Systems equipment

Content







Recommended minimum quantities for class of 30 pupils working in groups of three:

- **10 LEGO 1090 Sets**
- **10 Manual controllers 1039**
- **5 LEGO Interfaces 9750**
- **5 Computer systems**
- **5 Working disks**
- **5 Apple slotcard cable kits 9767**

However other combinations of sets and materials can be used very effectively.

Hints on use

Familiarity with materials

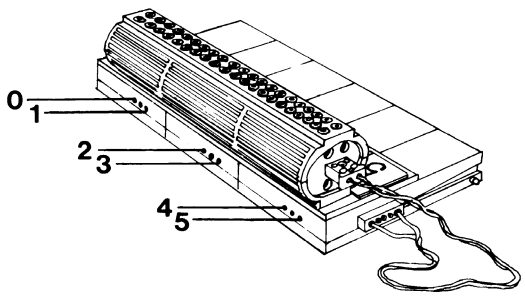
An introduction to the use of the equipment has been provided in the Teacher's materials.



Connecting the system

The computer system is connected as shown in the Resources booklet **R7**.

The manual controller



Storage and handling

Electronic equipment



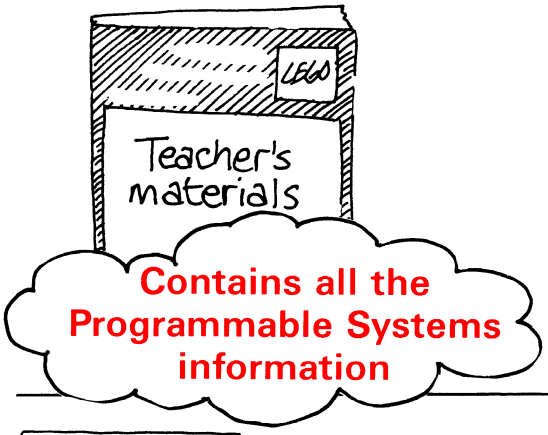
Connecting leads



Software disks

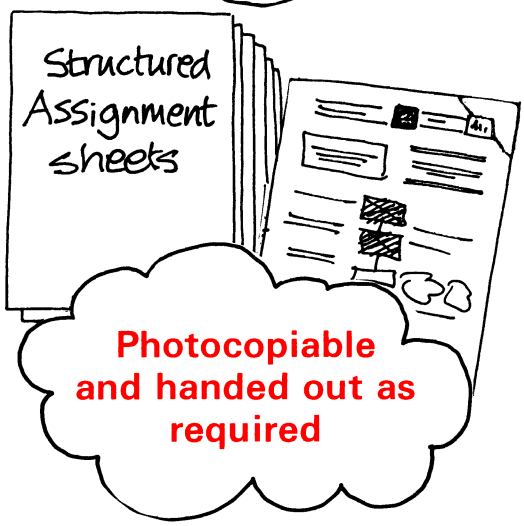


Programmable Systems resources

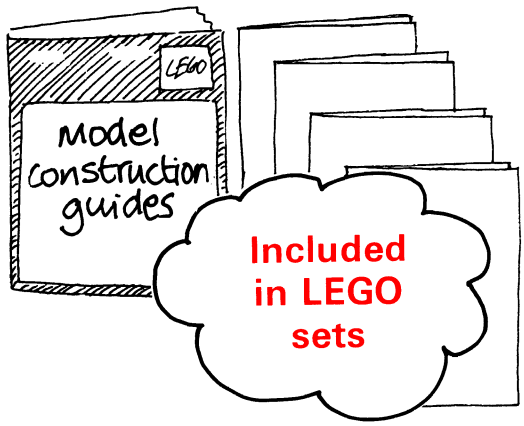


Contents

1 Introduction
2 Classroom materials
3 Resources booklet
4 Technical reference
5 Photocopy masters



A0 Introduction to materials (a–c)
A1 Introduction to Programmable systems (a–b)
A2 Introduction to computer control (a–f)
A3 Introduction to manual control
A4 Sensing and feedback (a–b)
A5 Computer control (a–f)
A6 Manual control
A7 Solving a problem
A8 Designing a structured program
A9 Designing programmed systems (a–j)



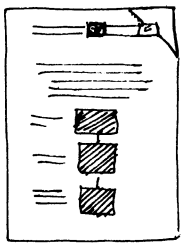
1090 A Ferris wheel
B Automatic door
C Washing machine
D Double conveyor
E Robot arm
1039 Manual controller

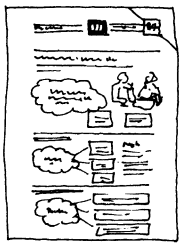
Recommended minimum quantities for class of 30 pupils working in groups of three:

Assignment sheets	**As required**	**Manual controller**	**1 per controller**
Construction guides	**1 per model**	**Keystrips**	
Resources booklets	**1 per group**	**Program sheets**	**As required**
		Assessment sheets	**As required**

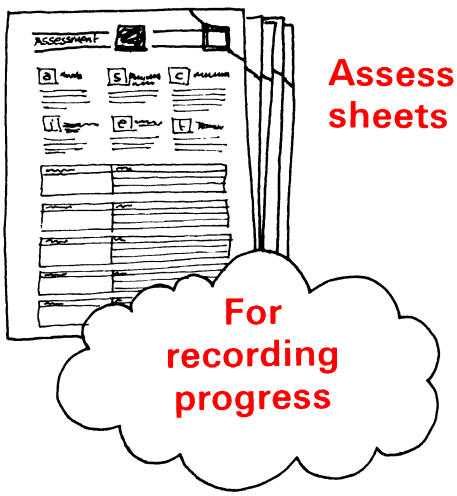


R0 Guide to contents
R1 Analysing the problem
R2 Finding and developing ideas
R3 Producing a solution
R4 Evaluating a solution
R5 Communicating your solution
R6 Manual controller
R7 Computer system
R8 LEGO interface — output
R9 LEGO interface — input
R10 LEGO *Lines* — reference
R11 LEGO *Lines* User Guide
R12 Feedback
R13 Developing a program
R14 Structured flow diagrams (SFD's)
R15 Vehicles — ideas for structures
R16 Vehicles — ideas for drive mechanisms
R17 Vehicles — ideas for steering mechanisms
R18 Vehicles — ideas for gearboxes
R19 Ideas — indoors
R20 Ideas — out of doors
R21 Ideas — at work
R22 Glossary of terms

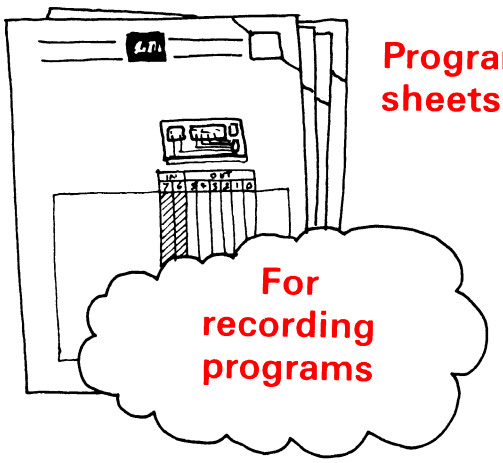








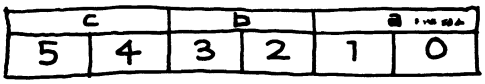
Assessment sheets



Program sheets

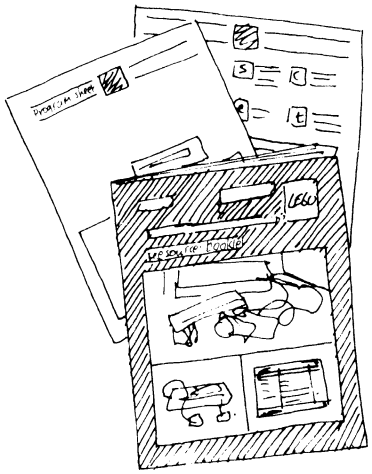
Keyboard strips

Manual controller



Guide to assignments

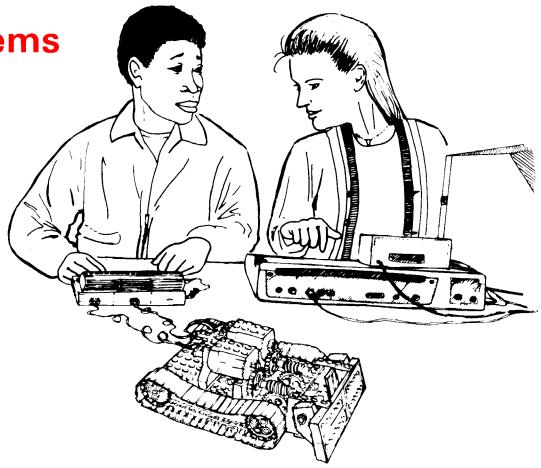
0 Introduction to the materials (page 2.8)



Gaining familiarity and confidence in the use of the materials.

Cards: A0 (a–c)
Models: 1090 D (recommended)
Software: CONVEY
Notes: Teacher-led demonstration of hardware and software; students explore resources on their own.

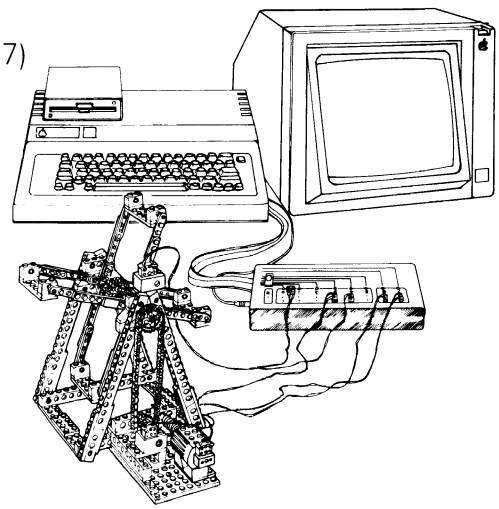
1 Introduction to Programmable Systems (page 2.13)



Controlling models using a computer and a manual controller.

Cards: A1 (a–b)
Models: 1090 A, 1090 B, 1090 D, 1090 E
Software: MLA (A1a only)
Notes: Models need to be built before lesson.

2 Introduction to computer control (page 2.17)



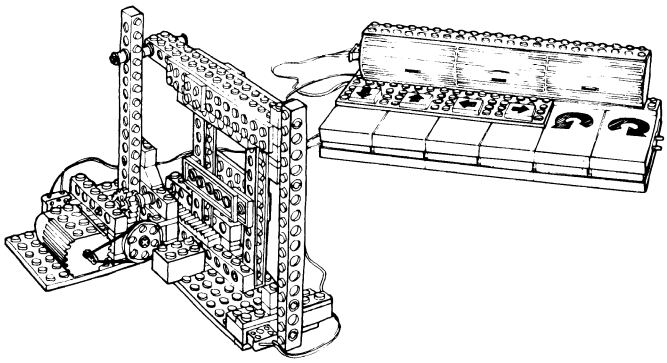
Learning how to control using software.

Cards: A2 (a–f)
Software: ONE to SIX
Notes: Motors and lights are extracted from models as required. An opto-sensor brick is also required.

3 Introduction to manual control (page 2.29)

Learning how to control using a mechanical system.

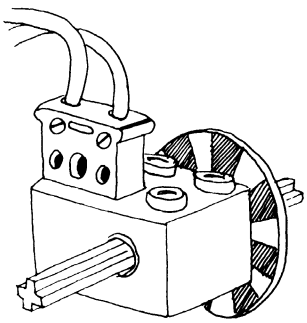
Card: A3
Notes: Gearbox units can be assembled from any of the construction guides (1090 B for example) but an example is supplied.



4 Sensing and feedback (page 2.31)

Learning how to use sensors.

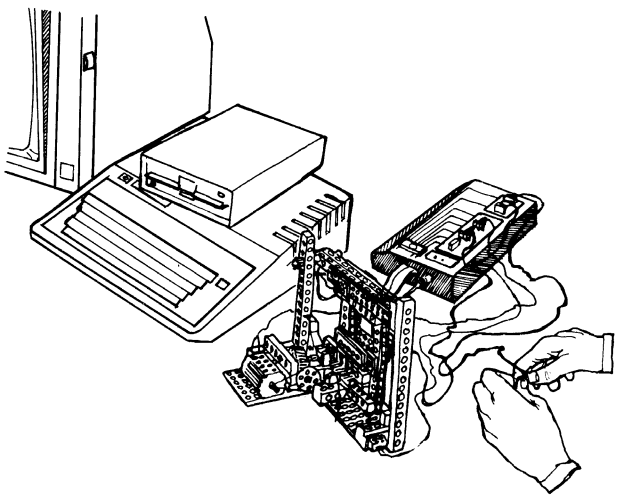
Cards: A4 (a–b)
Models: 1090 A, 1090 B, 1090 D, 1090 E
Software: MLB, COUNT
Notes: Motors and lights are extracted from models as required. An opto-sensor brick is also required.



5 Computer control (page 2.35)

Learning how to use sensors for computer control.

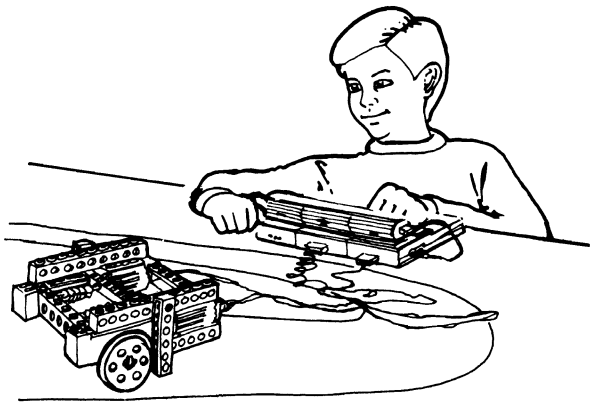
Cards: A5 (a–f)
Software: SEVEN to TWELVE
Notes: Motors, light bricks and opto-sensors are extracted from models and used to test sub-systems.



6 Manual control (page 2.47)

Learning about feedback through human control.

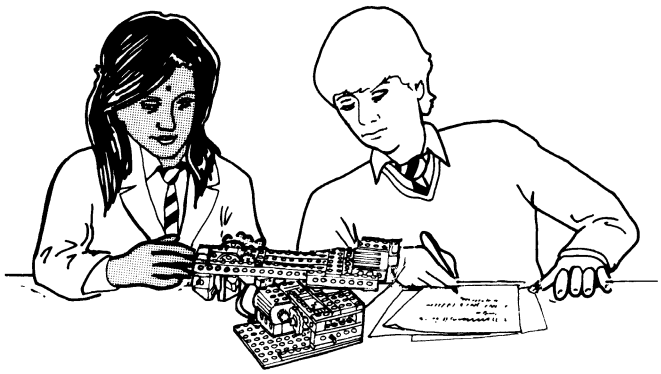
Card: A6
Model: A vehicle is built from the 1090 kit.
Notes: Models should be dismantled before and after this assignment.



7 Solving a problem (page 2.49)

Learning a structured approach to solving problems.

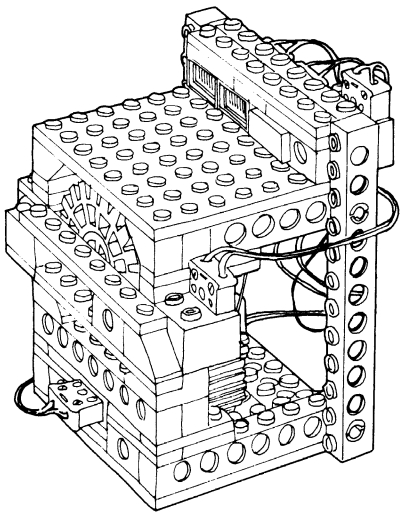
Card: A7
Notes: Teacher may wish to have a demonstration buggy already built, showing a solution to the problems of Assignment 6.



8 Designing a structured program (page 2.51)

Learning how to use the problem-solving process for computer control problems.

Card: A8
Model: 1090 C
Notes: 1090 C is constructed either before or during this assignment. There is no sample demonstration program for the problems posed.

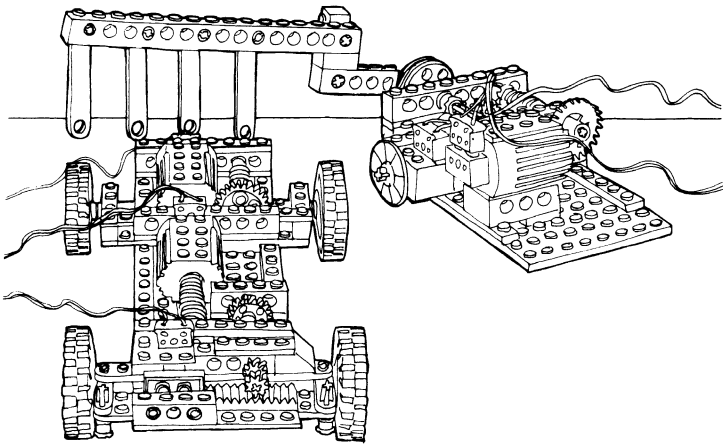


9 Designing programmed systems (page 2.53)

Applying the process to a computer control problem.

Cards: A9 (a–j)

Notes: A series of open-ended problems to be solved using all available resources.



Assignments 9a-9e use models which may be constructed from the construction guides. 9e is more complex than 9a.

Assignments 9f-9j are adaptations of existing models or require starting from scratch. 9j is the most complex.

The problems:

A9a A sliding door allowing customers in and out but otherwise remaining shut to conserve heat and energy.

A9b A safe and economic Ferris wheel ride.

A9c A device to pick LEGO bricks from a conveyor and deposit them on a platform nearby.

A9d A mobile space vehicle, capable of avoiding obstacles in its path.

A9e A conveyor system which can sort large boxes from small ones.

A9f A safe and efficient automatic barrier for a car park.

A9g A set of curtains which open and close automatically.

A9h An automatically controlled roundabout.

A9i A device to stamp a pattern on to a plasticine block.

A9j A theatrical head which comes alive as members of the cast pass it.



Purpose: Assignment 0

To introduce the students to the resources available.

Aims

Skill To introduce the use of:
- — Resources booklet
- — Program sheet
- — Assignment sheet
- — Assessment sheet
- — Hardware and software

Attitude To develop confidence in the use of these materials.

Resources (each group)

Resources booklet
Program sheet for each student
Assignment sheet for each student
One-motor model (1090 B)
Two-motor model (1090 E)
Leads
Manual controller (without batteries)
Interface

Resources (teacher)

Computer system
LEGO ***Lines*** program disk with Sample Programs

Notes

Teachers should ensure that the students are familiar with the materials before continuing with the other assignments. This will be achieved by this teacher-led assignment and a demonstration of the LEGO ***Lines*** software.

A suitable demonstration could be based on the use of the double conveyor (1090 D) which introduces many of the key concepts such as systems, feedback, sensing, processing and programming. Discussion could be encouraged which explores why such a machine might be useful and how the various bits (sub-systems) go to make up the whole machine. At this stage, showing the computer system in operation is more important than appreciating how it works.

The Program sheet and Assessment sheet corners have been left blank for the student to write on some reminder (a program name, perhaps) of the activity being pursued.

The answers to the questions are readily found and can themselves be the subject of debate and agreement. This applies particularly to those questions whose answer is a definition.

Assignment: 0

Introduction to the materials

A0a

You will need

Resources booklet
Program sheet
Assessment sheet

You are going to

Learn how to use the materials

Using the materials you are going to learn about three different ways to control devices:

— human control
— mechanical control
— electronic control

To do this, you will be given Assignment cards, like this one. This assignment, Assignment 0, comes in three parts, A0a, A0b and A0c.

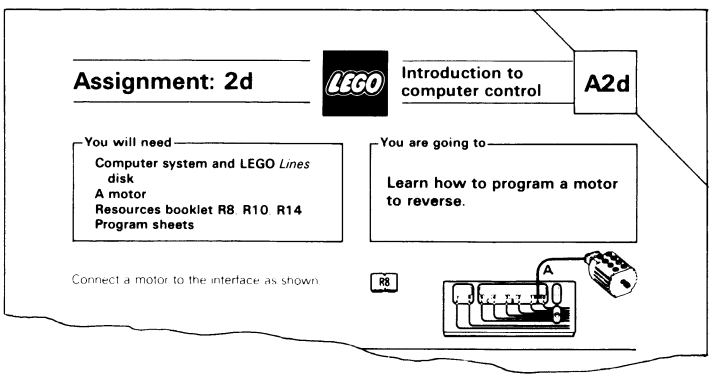
To help you in all the assignments, there are the Resources booklet, the Program sheet and the Assessment sheet. Make sure you have these in front of you for Assignment 0.

Assignment cards

The Assignment cards, like this one, are designed to be as easy to use as possible. On the top right-hand corner of each one you will find a reference number, A0a on this one (that is Assignment 0, part a). Underneath this you will find a statement of what you are going to do in the assignment.

The box on the top left-hand side of the card outlines what you will need in order to do the assignment. You should make sure you have these before you start.

The rest of the card tells you what to do. Drawings and sample programs take you through the activity. Where extra help is available in the Resources booklet, you are shown where to find it by the figure in the middle column like this: R8



Program sheet

Use resources page **R11** to discover what you will be writing in each of the columns.

© This material copyright LEGO Group, 1986. ®LEGO is a registered trade mark belonging to LEGO Group. This material may be freely photocopied *for use in the purchasing institution only.*

Assignment: 0

Introduction to the materials

A0b

You will need

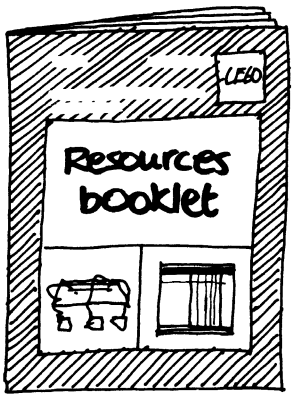
Resources booklet
Program sheet

You are going to

Explore the Resources booklet

Resources booklet

You will find that this booklet contains a great deal of information. The ***Guide to contents*** helps you to find what you want quickly and a ***Glossary of terms*** explains some of the words you will meet.



Use the Resources booklet to answer these questions.

1 What is the purpose of this booklet?

2 What is ***Technology***?

3 What resources would you use if you wanted help in designing control programs, using the LEGO materials?

4 What is the title of **R6**, and what kind of information do you find there?

5 If you were stuck because you didn't know what the term ***Cursor block*** means, where would you look for help? What is a Cursor block?

6 What is LEGO ***Lines***?

7 What are ***SFD***'s?

8 What does ***Evaluation*** mean?

9 If you had been told to produce a report, which resource page would you use?

10 Look at resource page **R10** and answer these questions:

What is a keyboard convention?

If you saw Press **Apple-L** in the text, what would you do?

Which keys are used to write labels?

Which keys are used to set output bits?

Which keys move the cursor block to the next line?

Which keys would you use to run a program?

Which key stops a program temporarily?

Which keys enable you to insert a new line?

Which keys test a single line of the program?

11 How would you connect a model with one light brick and one motor to the manual controller? How would you connect it to the interface (use the top of your Program sheet to answer). Do the same for a model with two motors and a light brick.

12 What is an output bit 4? An input bit 6? Output bits B?

© This material copyright LEGO Group, 1986. ®LEGO is a registered trade mark belonging to LEGO Group. This material may be freely photocopied *for use in the purchasing institution only.*

Assignment: 0

Introduction to the materials

A0c

You will need

Assessment sheet
Resources booklet

You are going to

Learn how to use the Assessment sheet

Assessment sheet

This is to help you measure the success of what you are doing and of what you have learnt. At the top there are headings for you to use to describe how you have got on, next to boxes containing letters (these letters also appear on the bottom of every Assignment sheet). Help is available on these headings on the following Resources booklet pages:

R1 R2 R3 R4 R5

1 What does **i** stand for?

2 Where would you look to find out more about what **a i s e c** and **t** stand for?

3 What are *Criteria*?

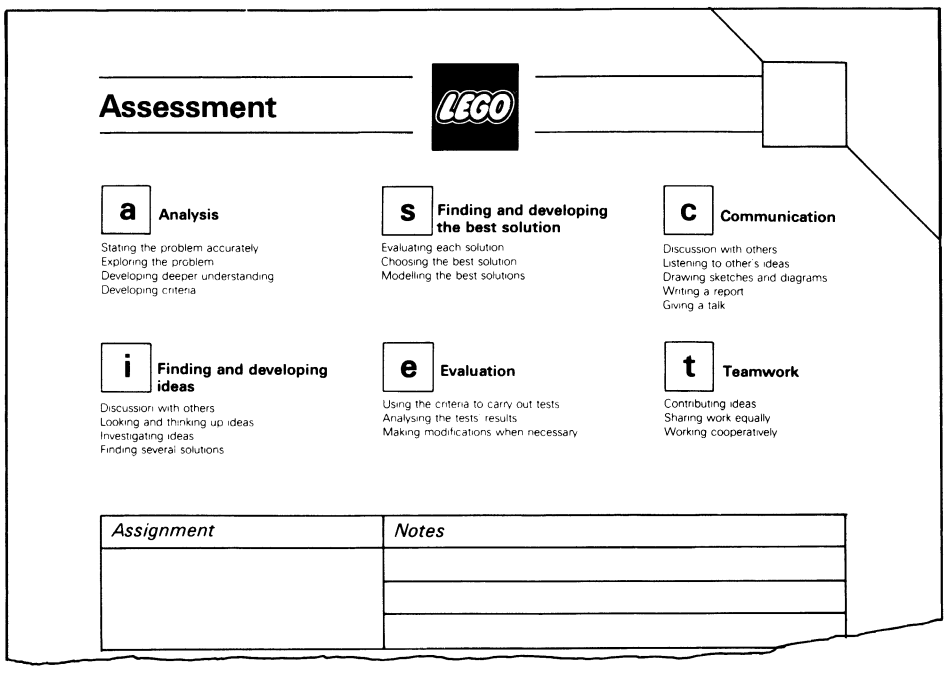
4 If you were asked how well you thought your group worked as a team, what questions would you ask yourself?

5 **c** has five elements in it. Which do you think will be the most difficult for you to achieve?

6 Do you think you will use all the items on the Assessment sheet in every single assignment you do?

7 Are there any terms you can't find out about on the Assessment sheet?

8 Fill in an Assessment sheet for Assignment 0, describing what you think you have learnt from answering these questions.



© This material copyright LEGO Group. 1986. ®LEGO is a registered trade mark belonging to LEGO Group. This material may be freely photocopied *for use in the purchasing institution only.*

Assignment: 1a

Purpose: Assignment 1

To control models using a computer and a manual controller.

Aims

Concept	Electrical devices can be controlled by computers.
Concept	A program consists of a series of instructions.
Skill	Manipulation of the manual controller to achieve a desired result.
Skill	Manipulation of a computer program to achieve a desired result.
Attitude	It is easy to alter the content of a control program in a software environment.

Resources (each group)

Computer system
 Computer, monitor and disk drive
Interface and power supply
LEGO *Lines* program disk
One-motor model with leads (1090 A or 1090 B)
Resources booklet **R7**, **R8**, **R11**, **R12**

Sample programs

`MLA`
`EGC`

Notes

Groups of three students (maximum).

This part of the assignment will occupy at least one half of a double lesson.

One half of the class may be working on this while the other works on assignment, **A1b**.

Each group should complete both parts of this assignment in the time allocated.

Each student in the group should use the computer keyboard and gain confidence in the ability to control a computer system.

If desired, the class could be taken through the activity by the teacher using a demonstration of a model working, OHPs and the LEGO *Lines* documentation.

This would reinforce the work covered in Assignment 0 and establish the use of the Resources booklet.

Extension work

Further instructions could increase the complexity of the programs. These should be discussed before being put into effect.

In particular, students should be encouraged to record and describe their solution.

A program listing is available as output to a printer, by pressing **`Apple-P`**, and this could form the basis of a description.

Troubleshooting

Keyboard problems. All students must establish control of the keyboard in this lesson.

All students should understand and use the following keystroke convention:

Press **`Key 1`**	Press key
Press **`Key 1 Key 2`**	Hold down first key, press other key, release first key

Program may have been altered by previous users or disk may be damaged.

Connections may be faulty or incorrectly set.

Model may not be a single motor model.

Disk drive or printer may not be standard.

Assignment: 1a

Introduction to programmable systems

A1a

You will need

Computer system with disk drive
LEGO *Lines* disk
1 motor model and light brick
Resource booklet

You are going to

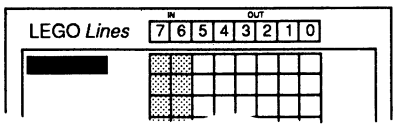
Use a computer program to control a model system.

Connect computer system and insert disk. R7

Collect model and connect it to the interface as directed in the Resources booklet. R8

Loading LEGO *Lines*

Turn the computer on. R10



Loading a program

Press **Apple-L** and wait for the list of programs to appear.
Move cursor, using arrow keys, to select MLA.
Press **RETURN** and wait for a few moments.

This program is designed to turn a light on, rotate a motor one way for two seconds, turn the light on again and rotate the motor the other way for another two seconds.

Running the program

Press **Apple-R** to run the program — check that the system performs correctly. If not → check connections and then Try again

Changing the program

R10

Alter the time of rotation

Press ↓ then → and check that the cursor block has moved down and across

Press **DELETE** to remove the time on this line:

oneway ▒▒ 0 0 0 0 0 1 2.0

Enter a number between 1.0 and 3.5. Check to see it is entered correctly.

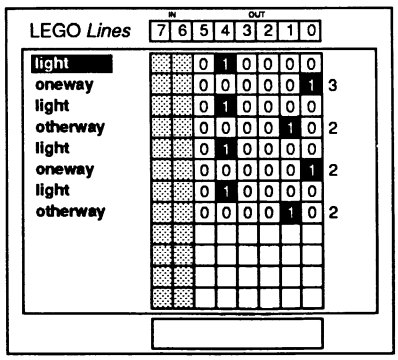
Press **Apple-R** to run the program. Then copy the program onto a Program sheet and describe what it does.

Enter more instructions to repeat the sequence

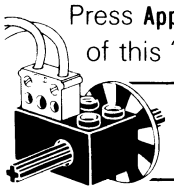
Press ↓ until the cursor block reaches the first blank line.

Copy the original instructions

Use: Letter keys for labels
Apple and number keys for output values
Numbers for timing or counting
RETURN to go to the next line. R10



Press **Apple-R** to run program and then produce a printout of this 'new' program and describe what it does.



a i s e c t

© This material copyright LEGO Group, 1986. ®LEGO is a registered trade mark belonging to LEGO Group. This material may be freely photocopied *for use in the purchasing institution only.*

Assignment: 1b

Purpose: Assignment 1

To control models using a computer and a manual controller.

Aims

Concept	Electrical devices can be controlled by computers.
Concept	A program consists of a series of instructions.
Skill	Manipulation of the manual controller to achieve a desired result.
Skill	Manipulation of a computer program to achieve a desired result.
Attitude	It is easy to alter the content of a control program in a software environment.

Resources (each group)

Manual controller (1039) with keystrip
Two-motor model with leads (1090 D or 1090 E)
Program sheets as required.
Batteries — spares

Notes

Groups of three (maximum).

This part of the assignment will occupy at least one half of a double lesson.

One half of the class may be working on this while the other works on assignment, **A1a**.

Each group should complete both parts of this assignment in the time allocated.

Each student in the group should use the manual controller and test a sequence of instructions they have written, preferably onto the Program sheet.

A description of each program in everyday English could accompany each set of instructions. Check that time values are being entered in the right hand column of the Program sheet.

Extension work

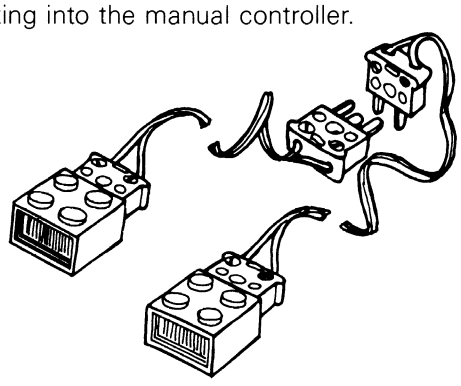
Repeat the assignment using a different two-motor model.

Troubleshooting

Two-motor model may not be connected correctly. Check **R8**.

The keyboard strip might not have been used to number keys correctly.

In this example, the lights should be connected by plugging one lead into the other before inserting into the manual controller.



Batteries may be dead — are extra ones available?

Assignment: 1b

Introduction to programmable systems

A1b

You will need

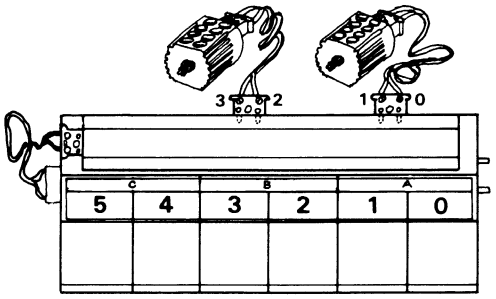
Manual controller and keystrip
Two-motor model (1090 D or 1090 E)
Resources booklet R6, R8
Program sheets

You are going to

Control a model using the manual controller.

Connect a two-motor model to the manual controller as shown in the Resources booklet.

R8



Press each key on the controller reasonably hard in turn.

Note what happens to the model.

A summary of your actions is given below:

The 1 means that key 1 is pressed or *on*

	5	4	3	2	1	0
Key 0	0	0	0	0	0	1
Key 1	0	0	0	0	1	0
Key 2						
Key 3						
Key 4						
Key 5						

Draw this table and complete it.

Try pressing keys 1 and 3 at the same time:

What happens to the model?

Why?

	5	4	3	2	1	0
Key 1 and Key 3	0	0	1	0	1	0

Check the direction the motors turn when you press keys 2 and 0 at the same time. If this is not the same, can you change the leads round so it is?

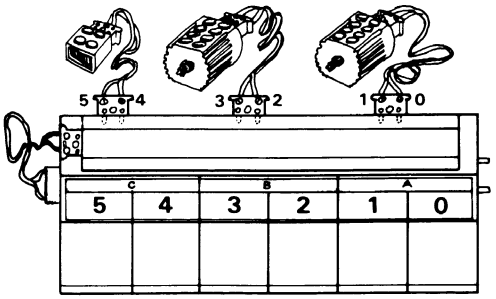
R8

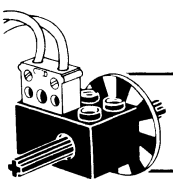
Here is a sequence of instructions.

	5	4	3	2	1	0	time
light on	0	1	0	0	0	0	5.0
one way	0	0	0	1	0	1	10.0
light on	0	1	0	0	0	0	5.0
other way	0	0	1	0	1	0	10.0

Instead of a key to press, we now have a ***label*** (to describe what is happening) and a time value to tell you how long to do it.

Now write some programs (on the program sheet) and try them out. Describe what you intend your models to do, and whether they do it.





a i s e c t

© This material copyright LEGO Group, 1986. ®LEGO is a registered trade mark belonging to LEGO Group. This material may be freely photocopied for use in the purchasing institution only.

Assignment: 2a

Purpose: Assignment 2

To learn the first set of commands which may be used in the LEGO *Lines* programming environment.

Aims

Concept	The output bit pattern is used to control lamps and motors.
Skill	The ability to design a simple sequence of instructions.
Attitude	It is straightforward to control motors and lamps using a computer system.

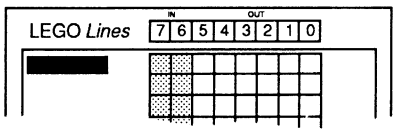
Resources (each group)

Computer system
 Computer, monitor and disk drive
Interface with power supply
LEGO *Lines* program disk
A motor and two light bricks with leads
Resources booklet **R7**, **R8**, **R10**, **R11**
Program sheets as required.

Sample program

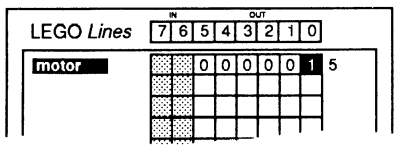
ONE

Put the disk into the disk drive and turn the computer on to load *Lines*.



Press **Apple-L** to load a *Lines* program.

Select **ONE** using arrow keys. Press **RETURN**.



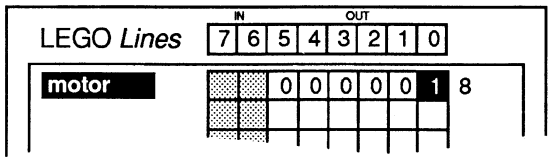
Notes

Groups of three students (maximum).

One half of the class may be working on this assignment while the other works on the complementary assignment, **A3**.

A group should complete this assignment in the time allocated.

If sufficient computers and interfaces are available, all groups could do this assignment simultaneously.



Assignments 2a–2f introduce software techniques and they must be completed in the set sequence.

The emphasis of this assignment is how instructions on screen are carried out through the interface. This relationship may need a demonstration from the teacher to be fully established.

It is important that students take the time to understand the six parts of this assignment. Rather than just rush ahead, they should describe what each program does either on a Program sheet or using a printout.

Extension work

If time permits, students could make up a further example of their own.

Bearing in mind the extent of this assignment, it is expected that students will be writing their own programs by the end of the session. **R13**

Troubleshooting

Check for bad connections between computer and interface.

Has the sequence of instructions on the assignment sheet been followed? (The program can be reloaded).

Check that the red STOP button on the interface is up.

Assignment: 2a

Introduction to computer control

A2a

You will need

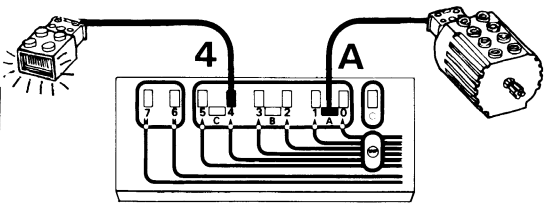
Computer system and LEGO *Lines* disk
A motor and a light brick
Resources booklet

You are going to

Learn how to control a motor and light brick.

Take a motor and a light brick from your model and connect them to the interface as shown.

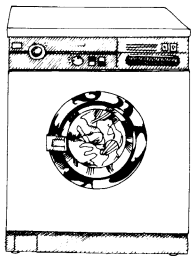
R8



A washing machine manufacturer needs a control program which will spin clothes at the end of the wash cycle for 5 seconds.

The LEGO motor represents the washing machine motor.

R8



Possible solution

Put the disk into the disk drive and turn the computer on.

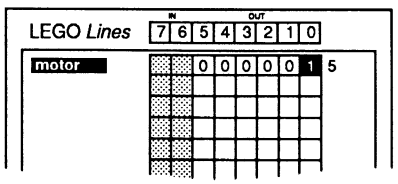
Connect motor to output bit 0 on the interface.

Press **Apple-L** to load program.

Select **ONE** with arrow keys. Press **RETURN**.

Press **Apple-R** to run program and describe what it does.

R10



An electric fan (*represented by the LEGO motor*) is required to come on for 8 seconds when a button is pressed.

Possible solution

Change time of last solution

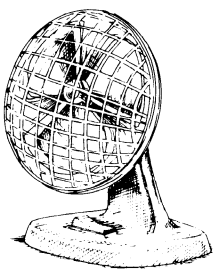
→ to move cursor block to 'time'

Press **DELETE**

Press **8** to enter new time.

Press **Apple-R** to run program and describe what it does — look especially at the interface.

R10



An outside light is required to come on for 8 seconds when a button is pressed.

Possible solution

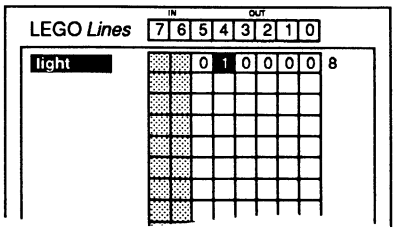
Set bit 0 off. (**Apple-0**)

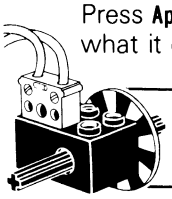
Set bit 4 on. (Look for a '1' in column 4). (**Apple-4**)

DELETE label and enter another word to describe what is taking place.

Press **Apple-R** to run program and describe what it does.

R10





a i s e c t

© This material copyright LEGO Group, 1986. ®LEGO is a registered trade mark belonging to LEGO Group. This material may be freely photocopied *for use in the purchasing institution only.*

Assignment: 2b

Purpose: Assignment 2

To learn the first set of commands which may be used in the LEGO *Lines* programming environment.

Aims

Concept The output bit pattern is used to control lamps and motors.

Skill The ability to design a simple sequence of instructions.

Attitude It is straightforward to control motors and lamps using a computer system.

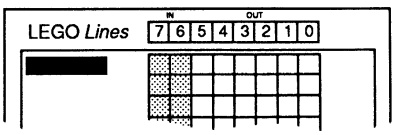
Resources (each group)

Computer system
 Computer, monitor and disk drive
Interface with power supply
LEGO *Lines* program disk
A motor and two light bricks with leads
Resources booklet **R7**, **R8**, **R10**, **R11**
Program sheets as required.

Sample program

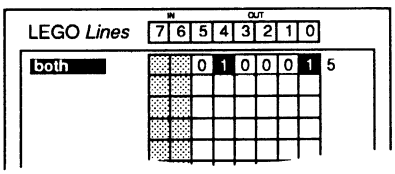
TWO

Load *Lines*.



Press **Apple-L** to load a *Lines* program.

Select and load program TWO.



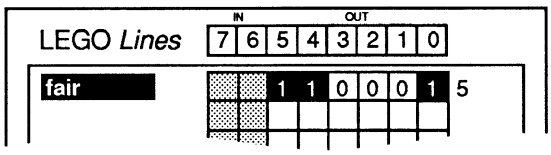
Notes

Groups of three students (maximum).

One half of the class may be working on this assignment while the other works on the complementary assignment, **A3**.

A group should complete this assignment in the time allocated.

If sufficient computers and interfaces are available, all groups could do this assignment simultaneously.



Assignments 2a–2f introduce software techniques and they must be completed in the set sequence.

The emphasis of this assignment is how instructions on screen are carried out through the interface. This relationship may need a demonstration from the teacher to be fully established.

It is important that students take the time to understand the six parts of this assignment. Rather than just rush ahead, they should describe what each program does either on a Program sheet or using a printout.

Extension work

If time permits, students could make up a further example of their own.

Bearing in mind the extent of this assignment, it is expected that students will be writing their own programs by the end of the session. **R13**

Troubleshooting

Check for bad connections between computer and interface.

Has the sequence of instructions on the assignment sheet been followed? (The program can be reloaded).

Check that the red STOP button on the interface is up.

Assignment: 2b

Introduction to computer control

A2b

You will need

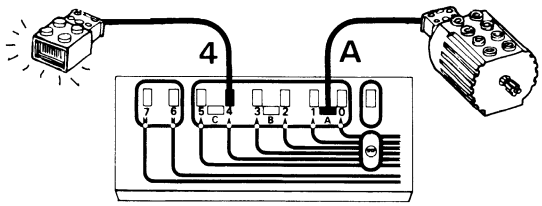
Computer system and LEGO *Lines* disk
A motor and two light bricks
Resources booklet R8, R10
Program sheets

You are going to

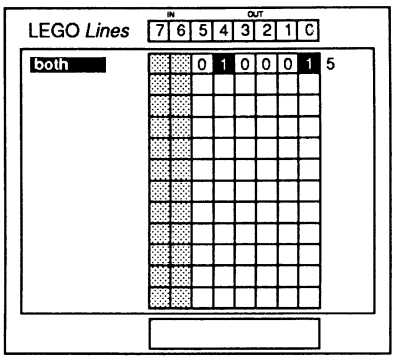
Learn how to control two or more output devices.

Use a light brick and a motor and connect them to the interface as shown.

R8



A program is required which operates a fairground ride (*represented by the LEGO motor*) and a light which shows that the ride is in progress for 5 seconds.



Possible solution

Connect motor to output bits A and lamp to output bit 4 on the interface.

R8

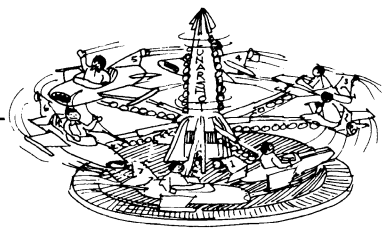
Press **Apple-L** to load program.

Select program TWO and load.

Press **Apple-R** to run program.

R10

Describe what happens on screen, on the interface and to the motor and light.



Improve the fairground ride to include two lights.

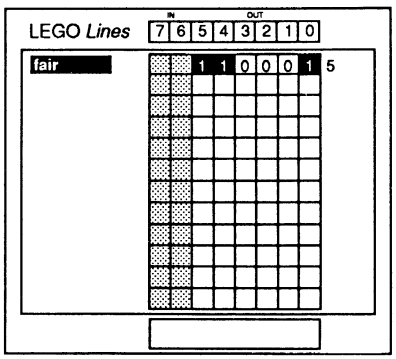
Possible solution

Connect another light brick to the interface.

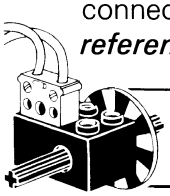
Change the line so that both the lights and the motor are on for 5 seconds.

Press **Apple-R** to run program.

R10



Make a record of your program on the Program sheet, obtain a printout of it (by pressing **Apple-P**) and describe how it solves the problem. You must have a printer connected to do this. See ***Technical reference*** section, page 4.11.



© This material copyright LEGO Group, 1986. ®LEGO is a registered trade mark belonging to LEGO Group. This material may be freely photocopied *for use in the purchasing institution only.*

Assignment: 2c

Purpose: Assignment 2

To learn the first set of commands which may be used in the LEGO *Lines* programming environment.

Aims

Concept The output bit pattern is used to control lamps and motors.

Skill The ability to design a simple sequence of instructions.

Attitude It is straightforward to control motors and lamps using a computer system.

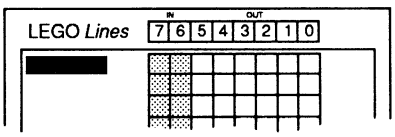
Resources (each group)

Computer system
 Computer, monitor and disk drive
Interface with power supply
LEGO *Lines* program disk
A motor and two light bricks with leads
Resources booklet **R7**, **R8**, **R10**, **R11**
Program sheets as required.

Sample program

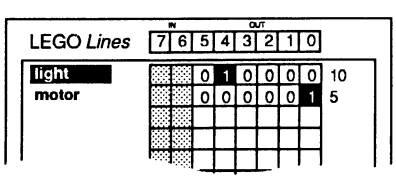
THREE

Load *Lines*.



Press **Apple-L** to load a *Lines* program.

Select and load program THREE.



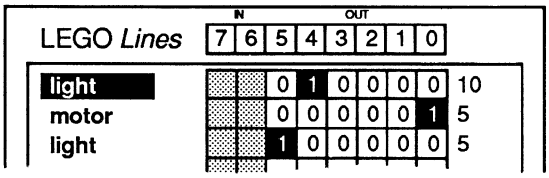
Notes

Groups of three students (maximum).

One half of the class may be working on this assignment while the other works on the complementary assignment, **A3**.

A group should complete this assignment in the time allocated.

If sufficient computers and interfaces are available, all groups could do this assignment simultaneously.



Assignments 2a–2f introduce software techniques and they must be completed in the set sequence.

The emphasis of this assignment is how instructions on screen are carried out through the interface. This relationship may need a demonstration from the teacher to be fully established.

It is important that students take the time to understand the six parts of this assignment. Rather than just rush ahead, they should describe what each program does either on a Program sheet or using a printout.

Extension work

If time permits, students could make up a further example of their own.

Bearing in mind the extent of this assignment, it is expected that students will be writing their own programs by the end of the session. **R13**

Troubleshooting

Check for bad connections between computer and interface.

Has the sequence of instructions on the assignment sheet been followed? (The program can be reloaded).

Check that the red STOP button on the interface is up.

Assignment: 2c

Introduction to computer control

A2c

You will need

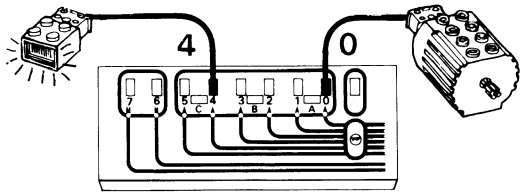
Computer system and LEGO *Lines* disk
A motor and two light bricks
Resources booklet R8, R10
Program sheets

You are going to

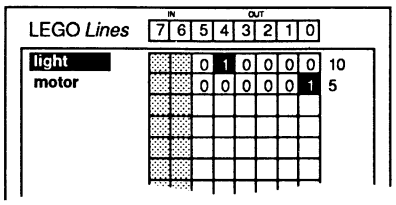
Learn how to program a sequence of instructions.

Connect the light brick and the motor to the interface as shown.

R8



A program is needed which will turn on a light 10 seconds before a motor is started up.



Possible solution

Connect motor to output bit 0.

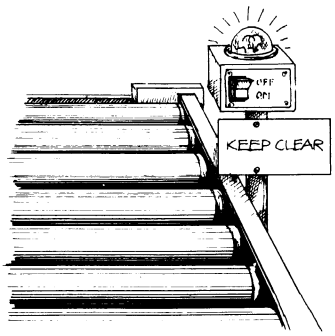
Connect lamp to output bit 4.

Press **Apple-L** to load a program.

Select and load program THREE.

Press **Apple-R** to run program and describe what happens.

R10



Modify the solution to turn on another lamp for 5 seconds after the motor has been turned off.

Possible solution

Connect another light brick to the interface.

R8

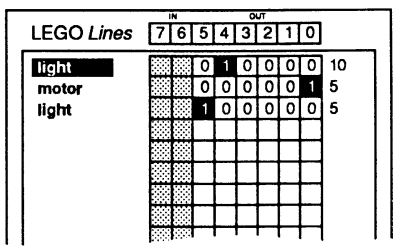
Modify the program by adding an extra line.

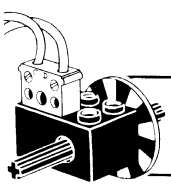
Position the cursor block at the start of this line and test it by pressing **Apple-T**.

Run the new program by pressing **Apple-R**.

R10

Make a record of your program on the Program sheet, obtain a printout of it (by pressing **Apple-P**) and describe how it solves the problem. You must have a printer connected to do this.





© This material copyright LEGO Group, 1986. ®LEGO is a registered trade mark belonging to LEGO Group. This material may be freely photocopied *for use in the purchasing institution only.*

Assignment: 2d

Purpose: Assignment 2

To learn the first set of commands which may be used in the LEGO *Lines* programming environment.

Aims

Concept	The output bit pattern is used to control lamps and motors.
Skill	The ability to design a simple sequence of instructions.
Attitude	It is straightforward to control motors and lamps using a computer system.

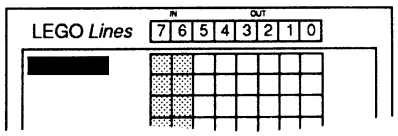
Resources (each group)

Computer system
 Computer, monitor and disk drive
Interface with power supply
LEGO *Lines* program disk
A motor with leads
Resources booklet **R7**, **R8**, **R10**, **R11**, **R14**
Program sheets as required.

Sample program

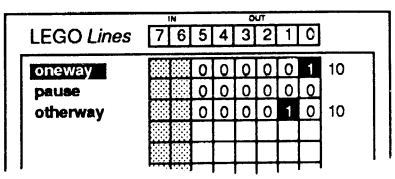
FOUR

Load *Lines*.



Press **Apple-L** to load a *Lines* program.

Select and load program FOUR.



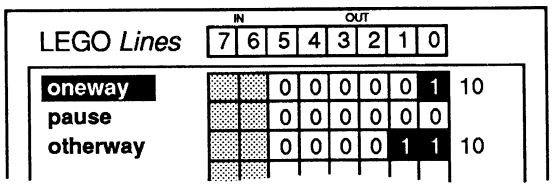
Notes

Groups of three students (maximum).

One half of the class may be working on this assignment while the other works on the complementary assignment, **A3**.

A group should complete this assignment in the time allocated.

If sufficient computers and interfaces are available, all groups could do this assignment simultaneously.



Assignments 2a–2f introduce software techniques and they must be completed in the set sequence.

The emphasis of this assignment is how instructions on screen are carried out through the interface. This relationship may need a demonstration from the teacher to be fully established.

It is important that students take the time to understand the six parts of this assignment. Rather than just rush ahead, they should describe what each program does either on a Program sheet or using a printout.

Extension work

If time permits, students could make up a further example of their own.

Bearing in mind the extent of this assignment, it is expected that students will be writing their own programs by the end of the session. **R13**

Troubleshooting

Check for bad connections between computer and interface.

Has the sequence of instructions on the assignment sheet been followed? (The program can be reloaded).

Check that the red STOP button on the interface is up.

Assignment: 2d

Introduction to computer control

A2d

You will need

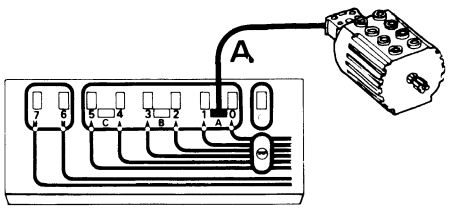
- **Computer system and LEGO *Lines* disk**
- **A motor**
- **Resources booklet R8, R10, R14**
- **Program sheets**

You are going to

Learn how to program a motor to reverse.

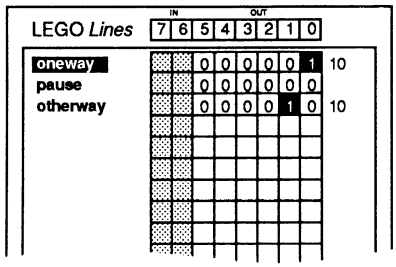
Connect a motor to the interface as shown.

R8



A program is needed to raise a barrier, pause and then lower the barrier again.

The solution should operate with simple button control (for example, **Apple-R***). The* LEGO *motor represents the barrier motor.*



Possible solution

Press **Apple-L** to load program.

Select and load FOUR.

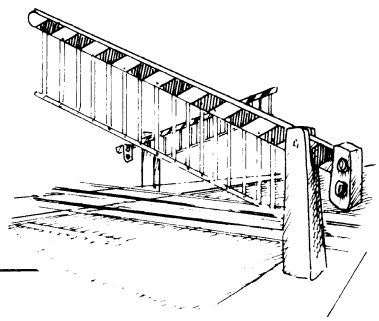
Press **Apple-R** to run program.

R10

Can you change the program so that the barriers pause for at least 15 seconds before starting again?

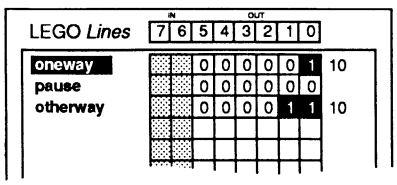
R14

Draw a structured flow diagram giving a solution to this problem and describe your solution.



Investigation

Change the last line of the program so that the output bits 0 and 1 are both on.



Press **Apple-T** to test new line.

R10

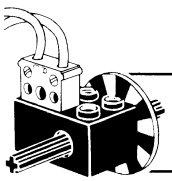
Explain what happens.

R19

Can you think of any everyday situations where this feature could be of any practical use?

R20

R21



© This material copyright LEGO Group, 1986. ®LEGO is a registered trade mark belonging to LEGO Group. This material may be freely photocopied *for use in the purchasing institution only.*

Assignment: 2e

Purpose: Assignment 2

To learn the first set of commands which may be used in the LEGO *Lines* programming environment.

Aims

Concept	The output bit pattern is used to control lamps and motors.
Skill	The ability to design a simple sequence of instructions.
Attitude	It is straightforward to control motors and lamps using a computer system.

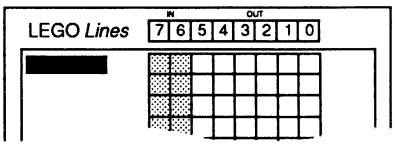
Resources (each group)

Computer system
 Computer, monitor and disk drive
Interface with power supply
LEGO *Lines* program disk
A motor and two light bricks with leads
Resources booklet **R7**, **R8**, **R10**, **R11**, **R13**
Program sheets as required.

Sample program

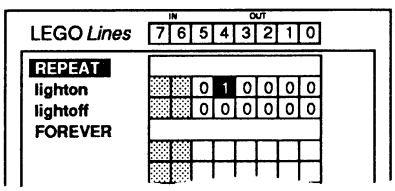
FIVE

Load *Lines*.



Press **Apple-L** to load a *Lines* program.

Select and load program FIVE.



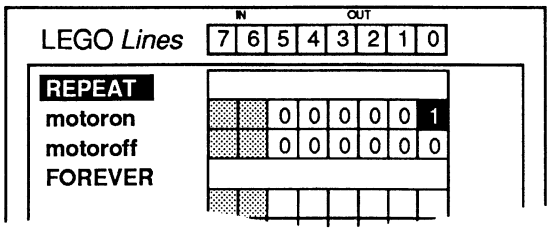
Notes

Groups of three students (maximum).

One half of the class may be working on this assignment while the other works on the complementary assignment, **A3**.

A group should complete this assignment in the time allocated.

If sufficient computers and interfaces are available, all groups could do this assignment simultaneously.



Assignments 2a–2f introduce software techniques and they must be completed in the set sequence.

The emphasis of this assignment is how instructions on screen are carried out through the interface. This relationship may need a demonstration from the teacher to be fully established.

It is important that students take the time to understand the six parts of this assignment. Rather than just rush ahead, they should describe what each program does either on a Program sheet or using a printout.

Extension work

If time permits, students could make up a further example of their own.

Bearing in mind the extent of this assignment, it is expected that students will be writing their own programs by the end of the session. **R13**

Troubleshooting

Check for bad connections between computer and interface.

Has the sequence of instructions on the assignment sheet been followed? (The program can be reloaded).

Check that the red STOP button on the interface is up.

Assignment: 2e

Introduction to computer control

A2e

You will need

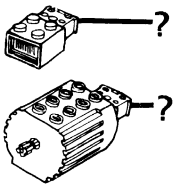
Computer system and LEGO *Lines* disk
A motor and two light bricks
Resources booklet R7, R8, R10, R11, R13
Program sheets

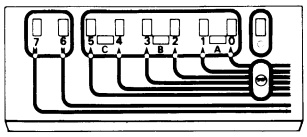
You are going to

Learn how to program a REPEAT FOREVER loop.

Use a motor and a light brick. You will have to work out from the programs where to connect them on the interface.

R8





A burglar alarm system requires a light which will flash on and of until a reset button is pressed.

Possible solution

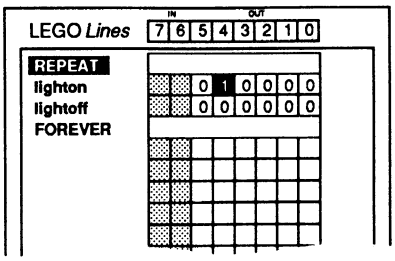
Press **Apple-L** to load program.

Select and load FIVE.

Connect the light brick to the correct output, once you have loaded the program.

Press **Apple-R** to run program.

R10



Investigation

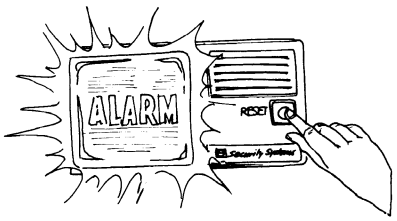
Press **SPACE** to halt program.

R10

Press **SPACE** to continue program.

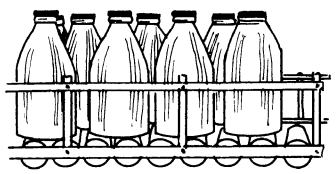
Press **ESCAPE** to end program.

Copy the instructions onto a Program sheet and describe what is happening.



Design a program to control a machine which presses tops onto milk bottles as they move along a continuous conveyor belt.

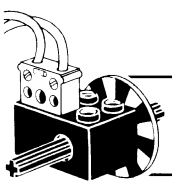
R13



What kind of device makes a conveyor move?
What kind of device can press a top onto a bottle?
How will the process of pressing the tops start and stop?

Use your answers to these questions to develop a solution to the problem using the LEGO materials.

Use the Program sheet to develop your solution and keep a record of your progress.



© This material copyright LEGO Group, 1986. ®LEGO is a registered trade mark belonging to LEGO Group. This material may be freely photocopied *for use in the purchasing institution only.*

Assignment: 2f

Purpose: Assignment 2

To learn the first set of commands which may be used in the LEGO *Lines* programming environment.

Aims

Concept The output bit pattern is used to control lamps and motors.

Skill The ability to design a simple sequence of instructions.

Attitude It is straightforward to control motors and lamps using a computer system.

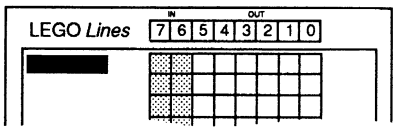
Resources (each group)

Computer system
 Computer, monitor and disk drive
Interface with power supply
LEGO *Lines* program disk
A motor and two light bricks with leads
Resources booklet **R7**, **R8**, **R10**, **R11**, **R13**
Program sheets as required.

Sample program

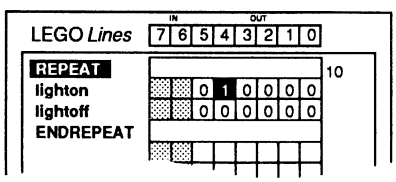
SIX

Load *Lines*.



Press **Apple-L** to load a *Lines* program.

Select and load program SIX.



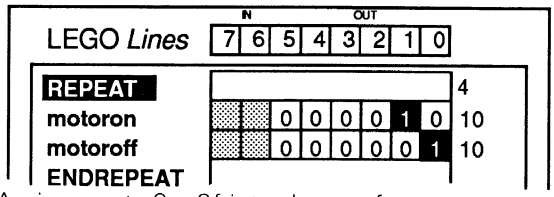
Notes

Groups of three students (maximum).

One half of the class may be working on this assignment while the other works on the complementary assignment, **A3**.

A group should complete this assignment in the time allocated.

If sufficient computers and interfaces are available, all groups could do this assignment simultaneously.



Assignments 2a–2f introduce software techniques and they must be completed in the set sequence.

The emphasis of this assignment is how instructions on screen are carried out through the interface. This relationship may need a demonstration from the teacher to be fully established.

It is important that students take the time to understand the six parts of this assignment. Rather than just rush ahead, they should describe what each program does either on a Program sheet or using a printout.

Extension work

If time permits, students could make up a further example of their own.

Bearing in mind the extent of this assignment, it is expected that students will be writing their own programs by the end of the session. **R13**

Troubleshooting

Check for bad connections between computer and interface.

Has the sequence of instructions on the assignment sheet been followed? (The program can be reloaded).

Check that the red STOP button on the interface is up.

Assignment: 2f

Introduction to computer control

A2f

You will need

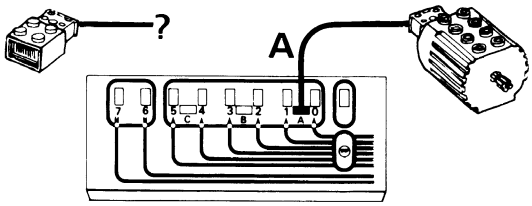
- **Computer system and LEGO *Lines* disk**
- **A motor and two light bricks**
- **Resources booklet R7, R8, R10, R11, R13**
- **Program sheets**

You are going to

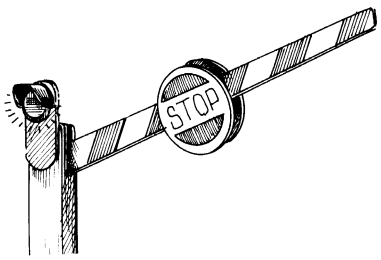
Learn how to program a REPEAT ENDREPEAT loop to run a set of instructions a number of times.

Use a motor and a light brick. Connect the motor to the interfaces as shown. You will have to work out from the programs where to connect the light brick.

R8



A level crossing barrier uses a flashing light to warn motorists it is closing. Design a program which will make this light flash 10 times.



Possible solution

Press **Apple-L** to load program.

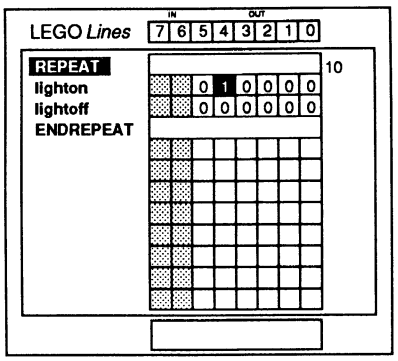
Select and load SIX.

Connect the light brick to the correct output, once you have loaded the program.

Press **Apple-R** to run program.

R10

Describe what happens in this program. How could the motor be included in this program?

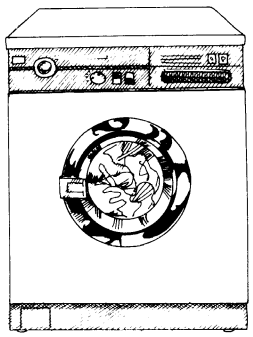


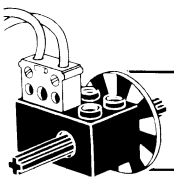
A washing machine manufacturer requires a control program which will rotate the drum one way for 10 seconds and then rotate it the opposite way for 10 seconds. This sequence needs to be repeated 4 times.

Using a light brick (*representing an on/off light*) and a LEGO motor (*representing the motor turning the drum*), write a program which solves this problem.

R13

Keep a record, on a Program sheet, of how your program develops.



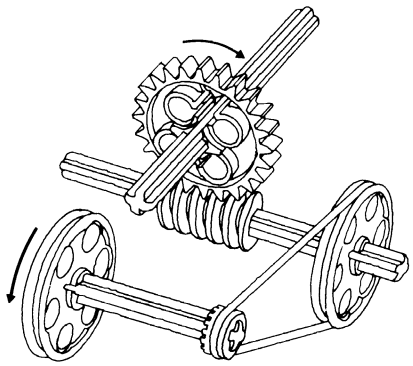


© This material copyright LEGO Group, 1986. ®LEGO is a registered trade mark belonging to LEGO Group. This material may be freely photocopied *for use in the purchasing institution only.*

Assignment: 3

Purpose: Assignment 3

To investigate mechanical control using a gearbox system.



Aims

Concept	Gears and pulleys can change the speed, directions and torque of rotating shafts.
Concept	Speed change is related to the relative number of teeth/ diameter of the gear or pulley system.
Skill	Ability to interpret and use information to solve simple problems.
Skill	Ability to apply mathematics to a practical problem.

Resources (each group)

LEGO model and leads
Construction Guide for model
Manual controller
Resources booklet **R6**, **R16–R21**
Stop clocks

Notes

Groups of three students (maximum).

One half of the class may be working on this assignment while the other works on the complementary assignment, **A2 (a–f)**.

A group should complete this assignment in the time allocated.

This assignment forms a very important basis for students' understanding of mechanical systems and mechanical control.

A record of the investigations in this assignment should be kept for future reference (especially for Assignments 6, 7 and 9).

This assignment requires a considerable amount of planning by the teacher. To avoid undue destruction of existing models, it would be desirable to be able to build gearboxes out of spare bits, or other LEGO Technic kits. A design is suggested on the student's assignment.

The output shaft speed of the gearbox should be:

speed of input motor × tooth ratio gear 1 × tooth ratio gear 2

and so on.

If a 16 tooth gear and a 24 tooth gear were used then the output shaft speed would be:

4000 × 1/16 × 1/24 rpm
= 10.42 rpm.

The actual output speed may differ significantly from this calculated speed because of friction and mechanical inertia. This may require some explanation.

Extension work

Identify everyday items which might make use of gearbox systems.

Investigate characteristics of:

Gear trains
Sprockets and chains (1090 A Ferris wheel)
Belt and pulley (1090 B Automatic door) (**R16–R18**)

Troubleshooting

Has the sequence of instructions on the assignment been followed?

Calculations of shaft speed may prove too difficult.

Assignment: 3

Introduction to manual control

A3

You will need

A model
A manual controller
Resources booklet R6, R16–21
Timing device (wristwatch)

You are going to

Learn how mechanical gears can control how quickly a motor turns a shaft.

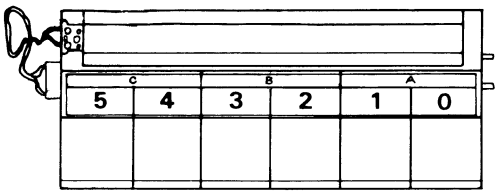
Connect the model to the manual controller.

R6

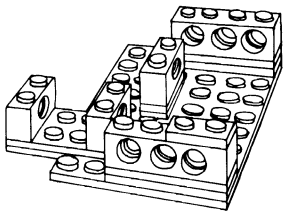
Make sure the connections are properly made.

Turn the motor on.

The model mechanism (the output shaft) turns more slowly than the motor which is driving it (the input shaft). Why should this happen?



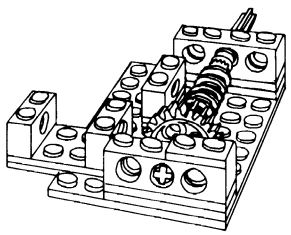
Build a gearbox like this one (you may need additional materials).

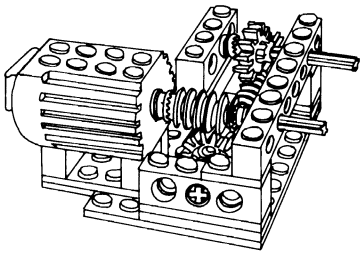


Why do the output shafts of this gearbox have different speeds to the motor?

R18

Discover if axles which turn quickly have a greater or smaller turning force than slower axles. Try slowing them down with your finger. Which is easier to stop?





Look at the section in the Resources booklet on gearboxes and drive mechanisms.

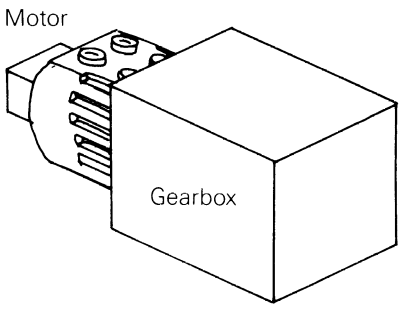
R16

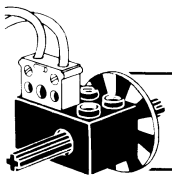
R18

How could you design a gearbox which will make your model move more slowly?

Use sketches and descriptions to record your ideas.

Build and test your design.





a i s e c t

© This material copyright LEGO Group, 1986. ®LEGO is a registered trade mark belonging to LEGO Group. This material may be freely photocopied *for use in the purchasing institution only.*

Assignment: 4a

Purpose: Assignment 4

To learn how to use the opto-sensor brick and to appreciate the limitations of human sensing and feedback.

Aims

Concept	The input of information needs a sensor.
Concept	Feedback is necessary for accurate control.
Attitude	It is easy to control a device using a computer system.

Resources (each group)

Computer system
 Computer, monitor and disk drive
Interface with power supply
LEGO *Lines* program disk
Opto-sensor brick with leads
Resources booklet **R7**, **R9**, **R10**, **R12**

IMPORTANT

When using COUNT, the line executed before COUNT will run continuously until the opto-sensor has counted the desired number of times. If the sample program shown is included in a **REPEAT** loop the 'motor' line will become the line ***before*** COUNT!

Inserting a line of 'zeros' after the motor line inside the **REPEAT** loop will stop the motor from running continuously.

Sample programs

```
MLB
COUNT
```

Notes

Groups of three students (maximum).

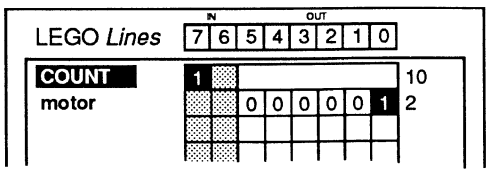
This part of the assignment should be completed in one half of the time allocated.

One half of the class may be working on this while the other works on **A4b**.

The opto-sensor brick responds to changes of light conditions. In the light, it could be either ON or OFF. This state can be 'flipped' by placing a finger over the sensor and removing it quickly. This feature should be explained to the students.

This assignment has been written so as to avoid stating the condition of the opto-sensor because it is indeterminate. The assignment is looking for ***change*** of light conditions.

The opto-sensor brick will work equally well in input bit 6 or 7 of the interface. They do not, of course, have any function if connected to any output bit.



Extension work

Gain further familiarity with input bits.

Explore uses of COUNT. In particular, this program could be used to give a more accurate measure of the gearbox speed calculated in Assignment 3.

Use the 4-segmented disc on the output shaft. Revise COUNT to count 400 sections on the disc.

Time this accurately (= 100 revs). Calculate number of revs per minute.

Troubleshooting

Check input bit connections and check one input bit is set to **any value** when using COUNT.

Check action of sensor.

Assignment: 4a

Sensing and feedback

A4a

You will need

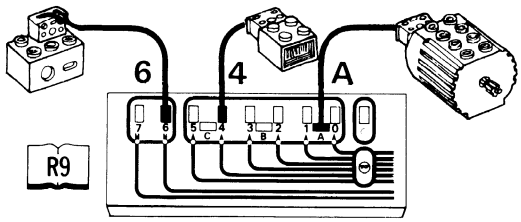
Computer system and LEGO *Lines* disk
A model with a light brick, a motor, and an opto-sensor brick
Resources booklet R7, R9–R12

You are going to

Learn how to use the opto-sensor brick.

Connect the motor on the model to output bits A.

Connect the light brick and opto-sensor to the interface as shown.

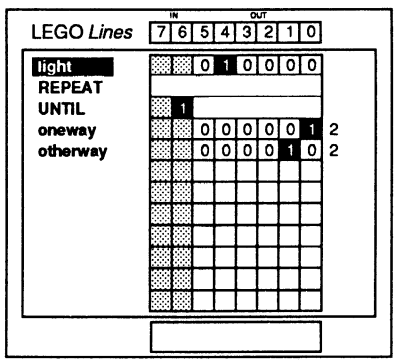


R9

On the interface bits 7 and 6 are input bits — that is, bits which let the computer system receive information from its environment.

Is the green indicator light for bit 6 on or off? Place your finger over the sensor slot — the indicator light will change. When you remove your finger it should return to its original state. Connect the opto-sensor to bit 7. Does it behave in the same way?

R10



Describe what you think is happening.

Reconnect the opto-sensor to bit 6 and connect it to your model. Connect the light brick to the model so that it shines into the opto-sensor, but 3 cms away from it.

Load the program called MLB. This program is designed to wait for a signal from the opto-sensor before switching the motor on for 2 seconds.

Press **Apple-R** to run the program and use your finger to trigger the sensor.

R9

Alter input sense

Pressing the arrow key, take the cursor down to the third line.

R11

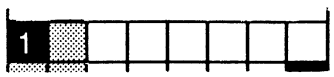
Press **Apple-6** to alter bit 6.

Press **Apple-R** to run the program again.

Can you explain why these two sets of instructions should behave differently?

Alter input line

Disconnect the opto-sensor from 6 and reconnect it to bit 7 on the interface.



▒ in bit 6 means 'any value'.

The sense of bit 6 will not matter when the UNTIL *looks for a signal.*

Press the arrow key to take the cursor to the third line.

Then press **Apple-6** to make bit 6 **any value**.

Press **Apple-7** to alter the value of bit 7 and press **Apple-R**.

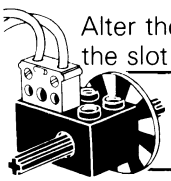
Counting input signals

R10

Load and run a program called **COUNT**.

This program waits for you to cover (and uncover) the sensor slot 10 times before continuing and turning the motor on.

Alter the program so that you only have to cover and uncover the slot 5 times before the motor is turned on.



© This material copyright LEGO Group, 1986. ®LEGO is a registered trade mark belonging to LEGO Group. This material may be freely photocopied *for use in the purchasing institution only.*

Assignment: 4b

Purpose: Assignment 4

To learn how to use the opto-sensor brick and to appreciate the limitations of human sensing and feedback.

Aims

Concept The input of information needs a sensor.

Attitude It is easy to control a device using a computer system.

Resources (each group)

LEGO model and leads
Manual controller
Stop clocks
Resources booklet **R6**, **R12**

Notes

Groups of three students (maximum).

This part of the assignment should be completed in one half of the time allocated.

One half of the class may be working on this while the other works on **A4a**.

Part A of this assignment shows that the human opto-sensor operates quite quickly. The lack of control in part B arises from the slowness of the feedback loop. This contrasts with the computer which provides a fast feedback system.

There are, however, other concepts at work here. The human system has considerable inertia associated with it and the actual speed of the signal between the various parts of the system is negligible when compared to this delay.

Students should be encouraged to describe the tasks set in this assignment.

Extension work

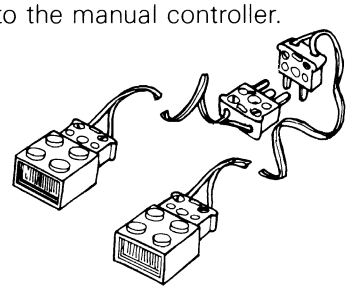
Students can be asked to look for other examples of feedback loops and to write descriptions of how they work.

Troubleshooting

The two-motor model may not be connected correctly.

The keyboard strip might not have been used to number keys correctly.

In this example, the lights should be connected by plugging one lead into the other before inserting into the manual controller.



Batteries may be dead — are extra ones available?

Assignment: 4b

Sensing and feedback

A4b

You will need

A model
Manual controller
Resources booklet R6, R12

You are going to

Learn about human sensing and feedback.

Part A

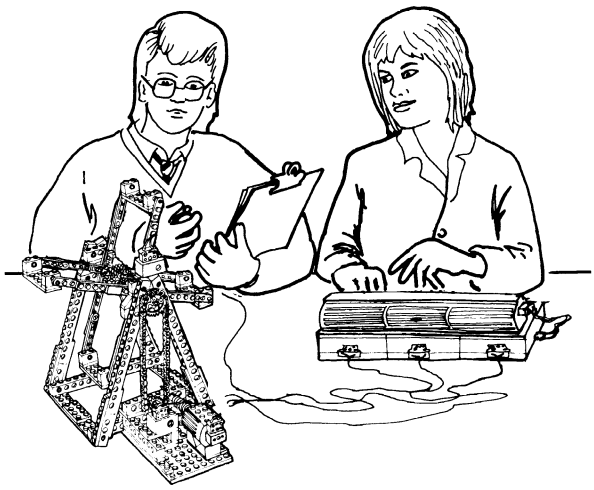
Decide on a task which fits your model (this should take about one minute to complete).

For example, use model 1090 A, the Ferris wheel.

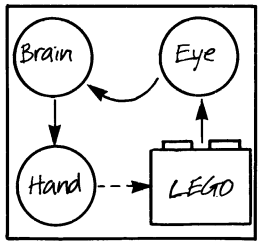
The task is to rotate the wheel so that each seat comes to rest at the bottom in turn so passengers can get on and off.

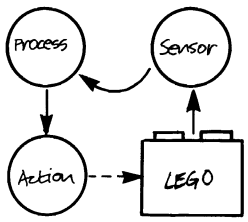
Use the manual controller and let each member of the group have a try. Make a note of the time it takes for each person to offload all the passengers.

Using the diagram, describe what is happening when a person performs this task (use the keywords ***sense***, ***process*** and ***act*** in your description).



R12





Part B

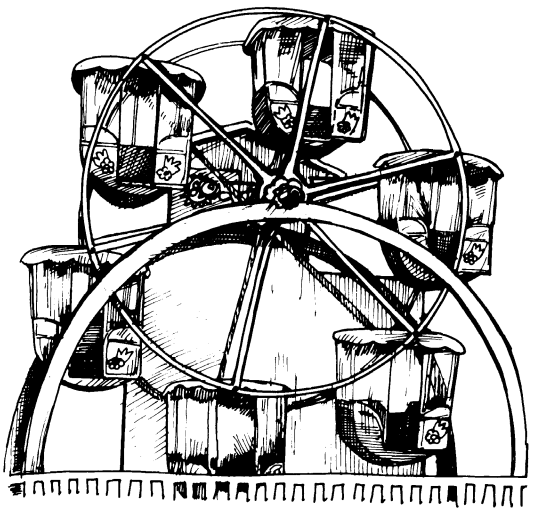
Whatever task you chose, you should now repeat it. This time, however, the person using the manual controller should have their eyes shut, so they cannot see the model.

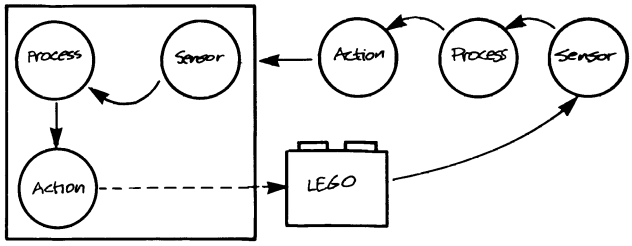
Another member of the group must now give instructions to the controller so the task can be performed. These might be 'Go forward', 'Back a bit' or 'Whoa!', whatever seems to work.

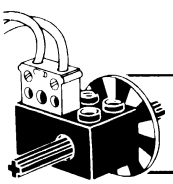
Again, you should make a note of the time it takes to complete the task.

How different were the times it took to complete the task?

Using the diagram, describe what happened (use the keyword ***feedback*** in your description). Try to explain why the tasks in part B took longer.







a i s e c t

© This material copyright LEGO Group, 1986. ®LEGO is a registered trade mark belonging to LEGO Group. This material may be freely photocopied *for use in the purchasing institution only.*

Assignment: 5a

Purpose: Assignment 5

To learn the second set of commands which may be used in the LEGO *Lines* programming environment.

Aims

Concept	Computers receive information through an input port.
Concept	Computers can be programmed to make decisions based on the input bit pattern.
Skill	The ability to design a simple sequence of instructions using the **REPEAT**, **UNTIL**, **IF**, **ENDIF** and **COUNT** keywords.
Attitude	It is straightforward to control devices using a computer system.

Resources (each group)

Computer system
 Computer, monitor and disk drive
Interface and power supply
LEGO *Lines* program disk
A motor and a light brick with leads
Two opto-sensor bricks with leads
Resources booklet **R7–R12**

Sample program

SEVEN

Notes

Groups of three students (maximum).

One half of the class may be working on this assignment while the other works on the complementary assignment, **A6**.

A group should complete this assignment in the time allocated.

If sufficient computers are available, all groups could do this assignment simultaneously (interfaces will be required).

Assignments 5a–5f introduce further software techniques and they must be completed in the set sequence.

Hard copies are required to document the exercises. Students should use the following:

- Program sheets
- Structured flow diagrams from printouts (**Apple-P**)
- Descriptions of each program.

From this point it is assumed that students know how to **LOAD** particular programs.

The focus of this assignment is on the new software commands. It is equally important that the relationship between what happens on the screen and on the interface is fully understood.

Extension work

Students can alter the set program.

Troubleshooting

Check that connections are not misplaced.

Difficulties may be experienced in understanding how the opto-sensor works.

In Assignment 5a the light brick is used to force an initial state but in other assignments 5b–5f it is recommended that students hold the sensor(s) so as to control its state.

Assignment: 5a

Computer control

A5a

You will need

- **Computer system and LEGO *Lines* disk**
- **A motor, a light brick and an opto-sensor brick**
- **Resources booklet R7–R10**

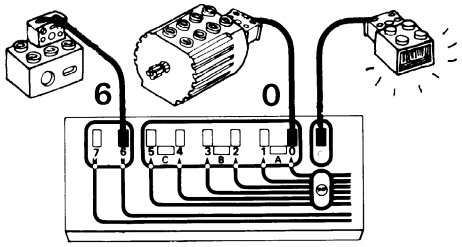
You are going to

Learn how to design a program which will REPEAT a sequence of instructions UNTIL an input message is sensed.

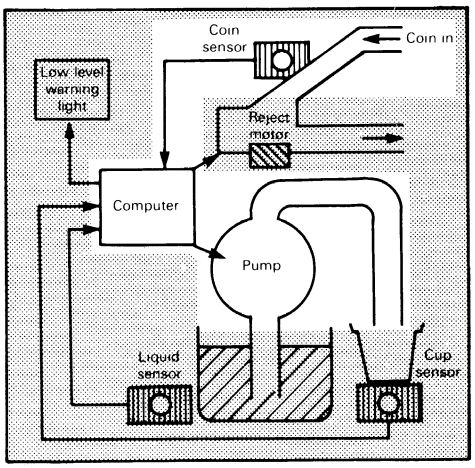
Connect the motor, the light brick and the opto-sensor brick to the interface as shown.

R8

R9



In a drinks dispensing system, a drink is to be pumped out only when a coin has been inserted. Design a program to solve this problem using the LEGO motor as the pump motor and the light brick and opto-sensor as a coin detector.

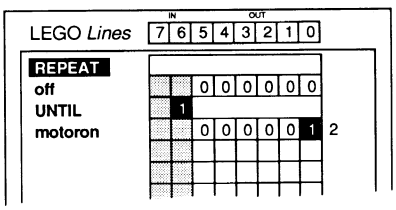


R9

Check that the opto-sensor is connected to input bit 6.

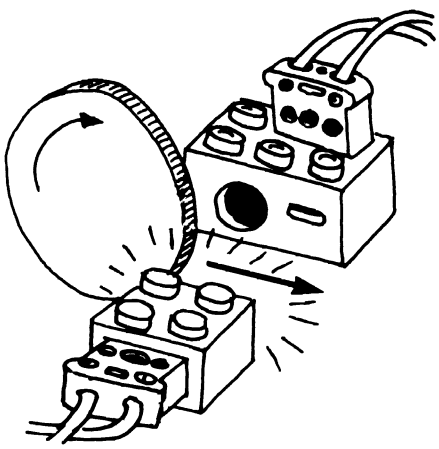
R8

Connect a light brick to the 4 v output supply and arrange it so that when the light shines into the opto-sensor the indicator light for bit 6 is off.

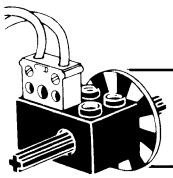


Load program **SEVEN**.

Press **Apple-R** to run the program. Cover the sensor to simulate a coin passing.



Use a hard copy of this program and describe what would happen in an actual drinks dispenser system once a coin had been detected.



© This material copyright LEGO Group, 1986. ®LEGO is a registered trade mark belonging to LEGO Group. This material may be freely photocopied *for use in the purchasing institution only.*

Assignment: 5b

Purpose: Assignment 5

To learn the second set of commands which may be used in the LEGO *Lines* programming environment.

Aims

Concept	Computers receive information through an input port.
Concept	Computers can be programmed to make decisions based on the input bit pattern.
Skill	The ability to design a simple sequence of instructions using the **REPEAT**, **UNTIL**, **IF**, **ENDIF** and **COUNT** keywords.
Attitude	It is straightforward to control devices using a computer system.

Resources (each group)

Computer system
 Computer, monitor and disk drive
Interface and power supply
LEGO *Lines* program disk
A motor and two light bricks with leads
Two opto-sensor bricks with leads
Resources booklet **R7–R12**

Sample program

EIGHT

Notes

Groups of three students (maximum).

One half of the class may be working on this assignment while the other works on the complementary assignment, **A6**.

A group should complete this assignment in the time allocated.

If sufficient computers are available, all groups could do this assignment simultaneously (interfaces will be required).

Assignments 5a–5f introduce further software techniques and they must be completed in the set sequence.

Hard copies are required to document the exercises. Students should use the following:

- Program sheets
- Structured flow diagrams from printouts (**Apple-P**)
- Descriptions of each program.

From this point it is assumed that students know how to **LOAD** particular programs.

The focus of this assignment is on the new software commands. It is equally important that the relationship between what happens on the screen and on the interface is fully understood.

Extension work

Students can alter the set program.

Troubleshooting

Check that connections are not misplaced.

Difficulties may be experienced in understanding how the opto-sensor works.

In Assignment 5a the light brick is used to force an initial state but in other assignments 5b–5f it is recommended that students hold the sensor(s) so as to control its state.

Assignment: 5b

Computer control

A5b

You will need

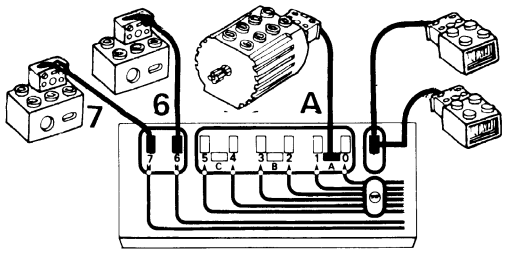
Computer system and LEGO *Lines* disk
A motor and two opto-sensor bricks and two light bricks
Resources booklet R7–R10

You are going to

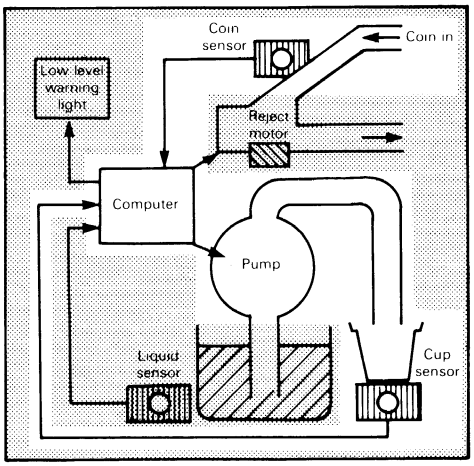
Learn how to design a program which requires two separate inputs to be sensed before an action is performed.

Connect the motor, the opto-sensors and the light bricks to the interface as shown.

R8-9



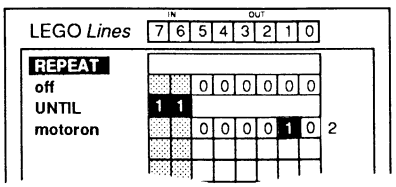
A program is required which will not dispense drinks until a coin is inserted *and* a cup is in place.



Connect the sensors to input bits 6 and 7. Make sure both indicator lights for bits 6 and 7 are off by placing your finger over the sensors as necessary, or by using the light brick as in **A5a**.

Connect the motor to output bits A.

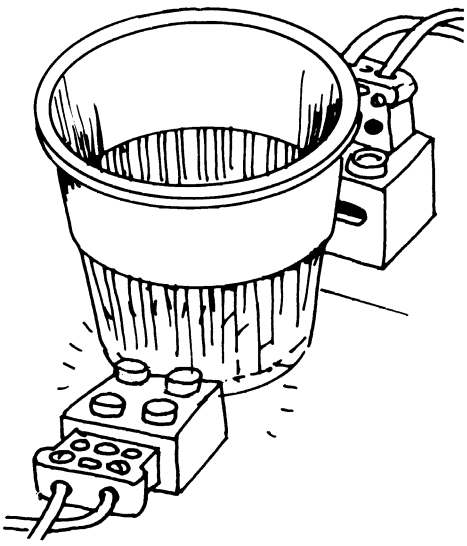
Load program **EIGHT**.

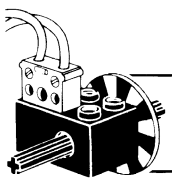


The opto-sensors are used to sense the presence of a coin and a cup.

Press **Apple-R** to run the program.

Use a hard copy of the program and describe what would happen in an actual drinks dispenser if the program were run.





a i s e c t

© This material copyright LEGO Group, 1986. ®LEGO is a registered trade mark belonging to LEGO Group. This material may be freely photocopied *for use in the purchasing institution only.*

Assignment: 5c

Purpose: Assignment 5

To learn the second set of commands which may be used in the LEGO *Lines* programming environment.

Aims

Concept	Computers receive information through an input port.
Concept	Computers can be programmed to make decisions based on the input bit pattern.
Skill	The ability to design a simple sequence of instructions using the **REPEAT**, **UNTIL**, **IF**, **ENDIF** and **COUNT** keywords.
Attitude	It is straightforward to control devices using a computer system.

Resources (each group)

Computer system
 Computer, monitor and disk drive
Interface and power supply
LEGO *Lines* program disk
A motor and an opto-sensor brick with leads
Resources booklet **R7–R12**

Sample program

NINE

Notes

Groups of three students (maximum).

One half of the class may be working on this assignment while the other works on the complementary assignment, **A6**.

A group should complete this assignment in the time allocated.

If sufficient computers are available, all groups could do this assignment simultaneously (interfaces will be required).

Assignments 5a–5f introduce further software techniques and they must be completed in the set sequence.

Hard copies are required to document the exercises. Students should use the following:

- Program sheets
- Structured flow diagrams from printouts (**Apple-P**)
- Descriptions of each program.

From this point it is assumed that students know how to **LOAD** particular programs.

The focus of this assignment is on the new software commands. It is equally important that the relationship between what happens on the screen and on the interface is fully understood.

Extension work

Students can alter the set program.

Troubleshooting

Check that connections are not misplaced.

Difficulties may be experienced in understanding how the opto-sensor works.

In Assignment 5a the light brick is used to force an initial state but in other assignments 5b–5f it is recommended that students hold the sensor(s) so as to control its state.

Assignment: 5c

Computer control

A5c

You will need

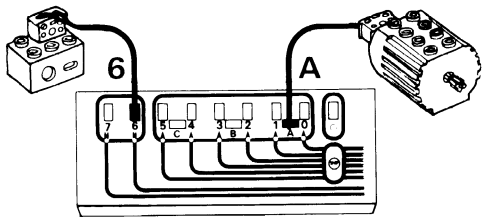
Computer system and LEGO *Lines* disk
A motor and an opto-sensor brick
Resources booklet R7–R10

You are going to

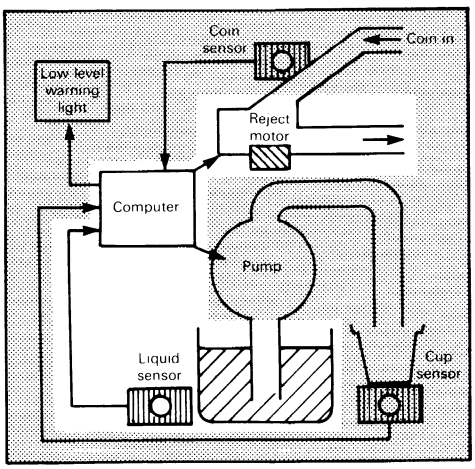
Learn how to design a program which will skip over a sequence of instructions until the right input is sensed.

Connect the motor and opto-sensor to the interface as shown.

R8-9



A program is required which will reject a coin if the liquid has been used up.



Condition the opto-sensor so that the indicator light on the interface is off at the start.

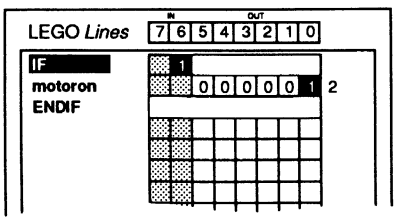
You could use a light brick (as before) or you could use your finger to activate the 'liquid sensor'.

The LEGO *motor represents the motor which rejects the coin.*

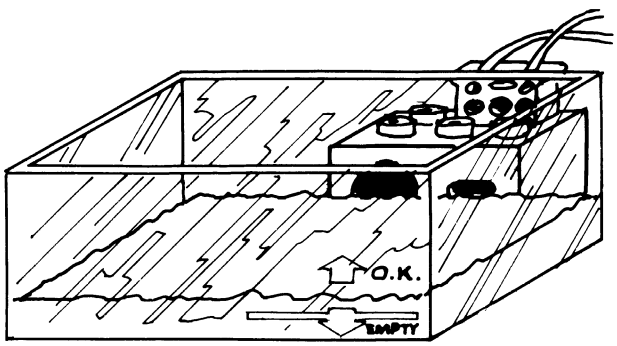
Load program **NINE**.

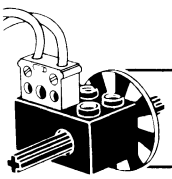
Press **Apple-R** to run the program.

Obtain a hard copy of the program and describe how it controls the motor.



Note that the motor does not come on (to reject the coin) until the sensor detects that the liquid has been used up.





a i s e c t

© This material copyright LEGO Group, 1986. ®LEGO is a registered trade mark belonging to LEGO Group. This material may be freely photocopied *for use in the purchasing institution only.*

Assignment: 5d

Purpose: Assignment 5

To learn the second set of commands which may be used in the LEGO *Lines* programming environment.

Aims

Concept	Computers receive information through an input port.
Concept	Computers can be programmed to make decisions based on the input bit pattern.
Skill	The ability to design a simple sequence of instructions using the **REPEAT**, **UNTIL**, **IF**, **ENDIF** and **COUNT** keywords.
Attitude	It is straightforward to control devices using a computer system.

Resources (each group)

Computer system
 Computer, monitor and disk drive
Interface and power supply
LEGO *Lines* program disk
A motor and two light bricks with leads
Two opto-sensor bricks with leads
Program sheets as required
Resources booklet **R7–R12**

Sample program

```
TEN
```

Notes

Groups of three students (maximum).

One half of the class may be working on this assignment while the other works on the complementary assignment, **A6**.

A group should complete this assignment in the time allocated.

If sufficient computers are available, all groups could do this assignment simultaneously (interfaces will be required).

Assignments 5a–5f introduce further software techniques and they must be completed in the set sequence.

Hard copies are required to document the exercises. Students should use the following:

- Program sheets
- Structured flow diagrams from printouts (**Apple-P**)
- Descriptions of each program.

From this point it is assumed that students know how to **LOAD** particular programs.

The focus of this assignment is on the new software commands. It is equally important that the relationship between what happens on the screen and on the interface is fully understood.

Extension work

Students can alter the set program.

Troubleshooting

Check that connections are not misplaced.

Difficulties may be experienced in understanding how the opto-sensor works.

In Assignment 5a the light brick is used to force an initial state but in other assignments 5b–5f it is recommended that students hold the sensor(s) so as to control its state.

Assignment: 5d

Computer control

A5d

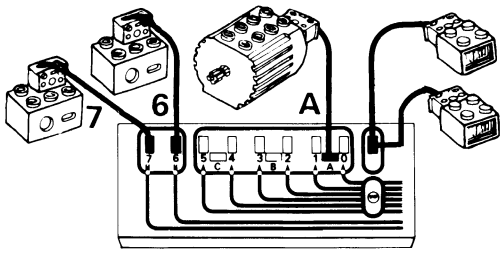
You will need

Computer system and LEGO *Lines* disk
A motor, two light bricks and two opto-sensor bricks
Resources booklet R7–R10

You are going to

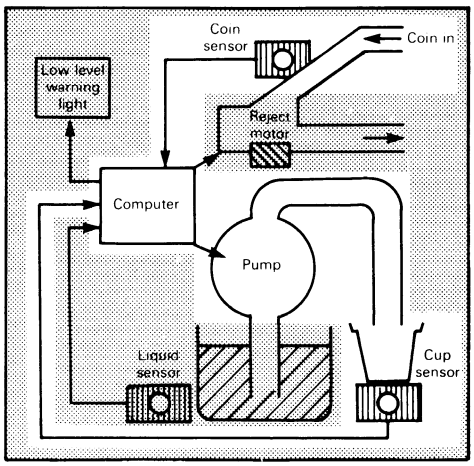
Learn how to design a program which will execute a sequence of instructions if two inputs are sensed.

Connect the light bricks, the motor and the opto-sensor bricks to the interface as shown.



A program is required which will only turn on the pump if a coin is inserted *and* a cup is in place.

Condition sensors so that the indicator lights on the interface are both off.

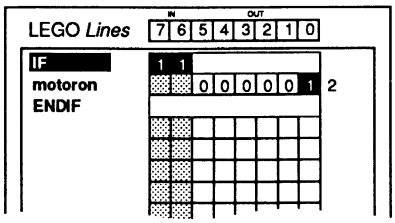


Load program **TEN**.

Press **Apple-R** to run the program.

Describe what happens.

Can you account for why the program 'appears' not to work?

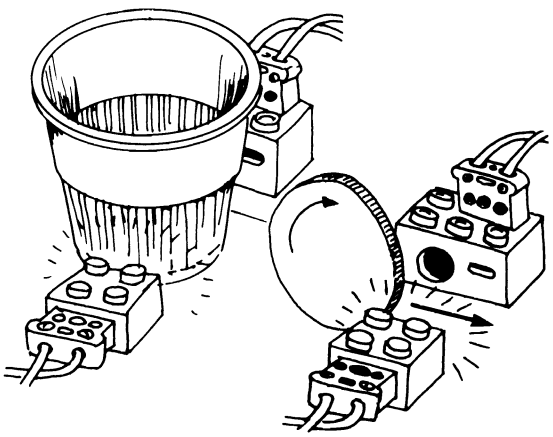


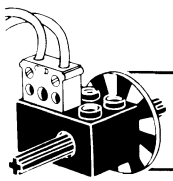
Note

The motor should ***not*** come on.

Condition the sensors so that the motor does come on.

Describe how you did this.





a i s e c t

© This material copyright LEGO Group, 1986. ®LEGO is a registered trade mark belonging to LEGO Group. This material may be freely photocopied *for use in the purchasing institution only.*

Assignment: 5e

Purpose: Assignment 5

To learn the second set of commands which may be used in the LEGO *Lines* programming environment.

Aims

Concept	Computers receive information through an input port.
Concept	Computers can be programmed to make decisions based on the input bit pattern.
Skill	The ability to design a simple sequence of instructions using the **REPEAT**, **UNTIL**, **IF**, **ENDIF** and **COUNT** keywords.
Attitude	It is straightforward to control devices using a computer system.

Resources (each group)

Computer system
 Computer, monitor and disk drive
Interface and power supply
LEGO *Lines* program disk
A motor and a light brick with leads
An opto-sensor brick with leads
Program sheets as required
Resources booklet **R7–R12**

Sample program

ELEVEN

Notes

Groups of three students (maximum).

One half of the class may be working on this assignment while the other works on the complementary assignment, **A6**.

A group should complete this assignment in the time allocated.

If sufficient computers are available, all groups could do this assignment simultaneously (interfaces will be required).

Assignments 5a–5f introduce further software techniques and they must be completed in the set sequence.

Hard copies are required to document the exercises. Students should use the following:

- Program sheets
- Structured flow diagrams from printouts (**Apple-P**)
- Descriptions of each program.

From this point it is assumed that students know how to **LOAD** particular programs.

The focus of this assignment is on the new software commands. It is equally important that the relationship between what happens on the screen and on the interface is fully understood.

Extension work

Students can alter the set program.

Troubleshooting

Check that connections are not misplaced.

Difficulties may be experienced in understanding how the opto-sensor works.

In Assignment 5a the light brick is used to force an initial state but in other assignments 5b–5f it is recommended that students hold the sensor(s) so as to control its state.

Assignment: 5e

Computer control

A5e

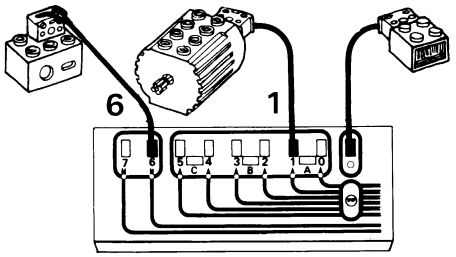
You will need

Computer system and LEGO *Lines* disk
A motor, a light brick and an opto-sensor brick
Resources booklet R7–R10

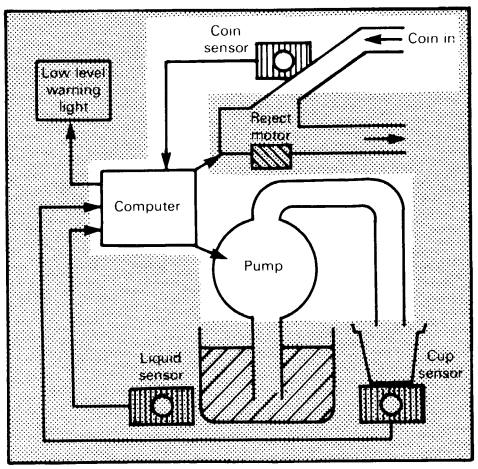
You are going to

Learn how to design a program which continually looks for an input and, when that input is sensed, executes a sequence of instructions.

Connect the motor, light brick and opto-sensor brick to the interface as shown.

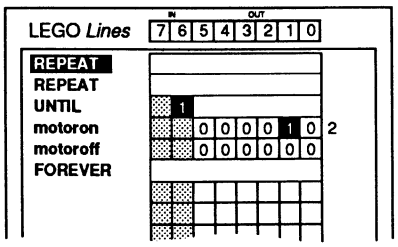


A program is required which continually looks for a coin and dispenses a drink when the coin is inserted.



Condition the sensor so that the indicator light on the interface is off.

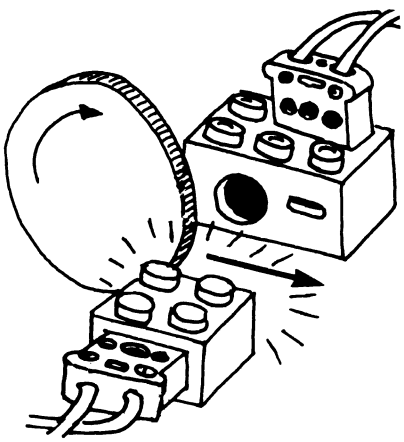
Load program **ELEVEN**.



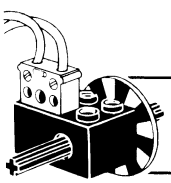
Press **Apple-R** to run the program.

When you condition the sensor to be on, the program will turn the motor on, then off. You can condition the sensor to be off again to simulate waiting for a coin to be inserted.

This will be repeated until the **ESCAPE** key is pressed.



Copy this program onto a Program sheet and explain how it differs from previous examples in this assignment.



a i s e c t

© This material copyright LEGO Group, 1986. ®LEGO is a registered trade mark belonging to LEGO Group. This material may be freely photocopied *for use in the purchasing institution only.*

Assignment: 5f

Purpose: Assignment 5

To learn the second set of commands which may be used in the LEGO *Lines* programming environment.

Aims

Concept	Computers receive information through an input port.
Concept	Computers can be programmed to make decisions based on the input bit pattern.
Skill	The ability to design a simple sequence of instructions using the **REPEAT**, **UNTIL**, **IF**, **ENDIF** and **COUNT** keywords.
Attitude	It is straightforward to control devices using a computer system.

Resources (each group)

Computer system
 Computer, monitor and disk drive
Interface and power supply
LEGO *Lines* program disk
A motor and two light bricks with leads
Two opto-sensor bricks with leads
Program sheets as required
Resources booklet **R7–R12**

Sample program

```
TWELVE
```

Notes

Groups of three students (maximum).

One half of the class may be working on this assignment while the other works on the complementary assignment, **A6**.

A group should complete this assignment in the time allocated.

If sufficient computers are available, all groups could do this assignment simultaneously (interfaces will be required).

Assignments 5a–5f introduce further software techniques and they must be completed in the set sequence.

Hard copies are required to document the exercises. Students should use the following:

- Program sheets
- Structured flow diagrams from printouts (**Apple-P**)
- Descriptions of each program.

From this point it is assumed that students know how to **LOAD** particular programs.

The focus of this assignment is on the new software commands. It is equally important that the relationship between what happens on the screen and on the interface is fully understood.

Extension work

Students can alter the set program.

Troubleshooting

Check that connections are not misplaced.

Difficulties may be experienced in understanding how the opto-sensor works.

In Assignment 5a the light brick is used to force an initial state but in other assignments 5b–5f it is recommended that students hold the sensor(s) so as to control its state.

Assignment: 5f

Computer control

A5f

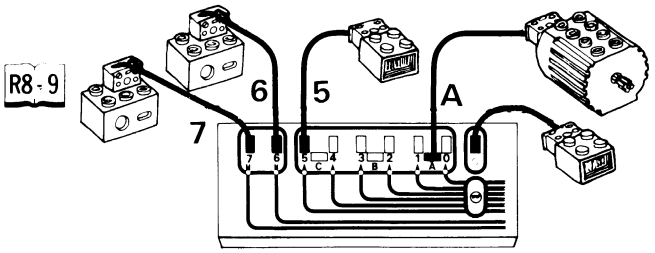
You will need

Computer system and LEGO *Lines* disk
A motor, two light bricks and two opto-sensor bricks
Resources booklet R7–R10

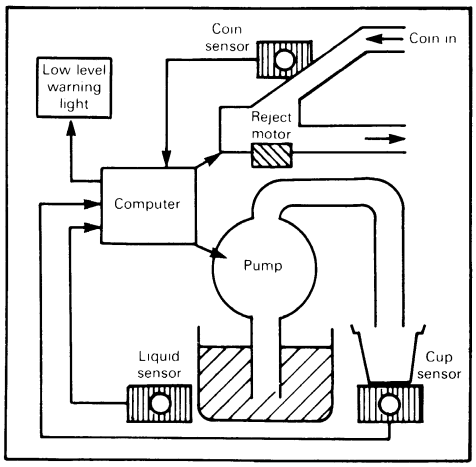
You are going to

Learn how to design a program which executes a sequence of instructions when an input is sensed. As part of those instructions another input will be tested and, if it is on, the appropriate action will be performed.

Connect the motor, light bricks and opto-sensor bricks to the interface as shown.



A program is required which looks for a coin and dispenses a drink when the coin is inserted. The program should then check the liquid level and, if it is too low, a warning light should be lit.

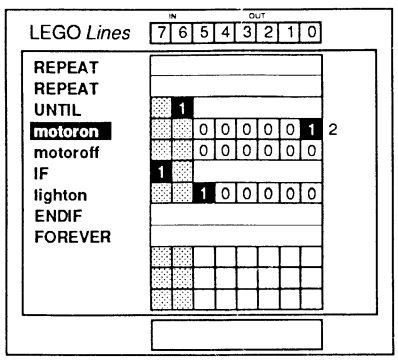


Condition the opto-sensors to be off.

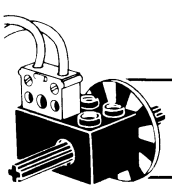
Load program **TWELVE**.

Press **Apple-R** to run the program.

ESCAPE will stop the program.



Copy this program onto a Program sheet and describe what would happen in an actual drinks dispenser if it were run.



a i s e c t

© This material copyright LEGO Group, 1986. ®LEGO is a registered trade mark belonging to LEGO Group. This material may be freely photocopied *for use in the purchasing institution only.*

Assignment: 6

Purpose: Assignment 6

To design and build a vehicle to be controlled by human feedback.

Aims

Concept	Accurate control is dependent on feedback actions being implemented quickly relative to the speed of reaction of the system itself.
Skill	The ability to design, test and evaluate a system.
Skill	The ability to build a simple sequence of instructions to control a device.
Attitude	The willingness and the confidence to solve problems.

Resources (each group)

A1 size paper (for drawing route)
LEGO 1090 Set (unassembled)
Manual controller
Resources booklet **R6**, **R15–R21**

Notes

A **teacher-led** activity for groups of three students.

One half of the class may be working on this assignment while the other works on the complementary assignment, **A5 (a–f)**.

A group should complete this assignment in the time allocated.

A possible approach to this assignment is to use the Resources booklet pages **R15–R17** to explore vehicle structures, drive mechanisms and methods of steering. Groups should be encouraged to explore ideas for themselves, select possible solutions and build prototypes for testing before creating their final model. A formal design session may be required (by both teachers and students) as part of this assignment.

Students should dismantle their vehicle at the end of the assignment, ready for the next group to start from scratch.

Troubleshooting

Check for efficient use of time.

Suitable vehicles are illustrated on the reverse of LEGO 1090 E Construction Guide or 1090 Series Guide.

Check for accurate record keeping (this is required for Assignment 7).

Assignment: 6

Manual control

A6

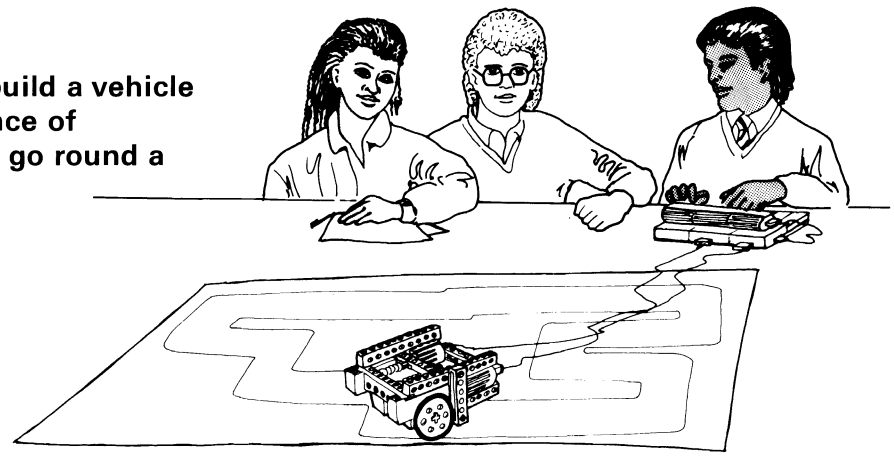
You will need

LEGO 1090 Kit
Manual controller
Resources booklet R6, R15–21
A1 sheet of paper

You are going to

Design and make a vehicle which you can then control.

Your group is required to build a vehicle and then to write a sequence of instructions to enable it to go round a course.



Building the vehicle

R15 shows some vehicle structures

R16 shows some drive mechanisms

R17 shows some steering mechanisms

Using these ideas, and ideas of your own, build the vehicle.

Draw a route

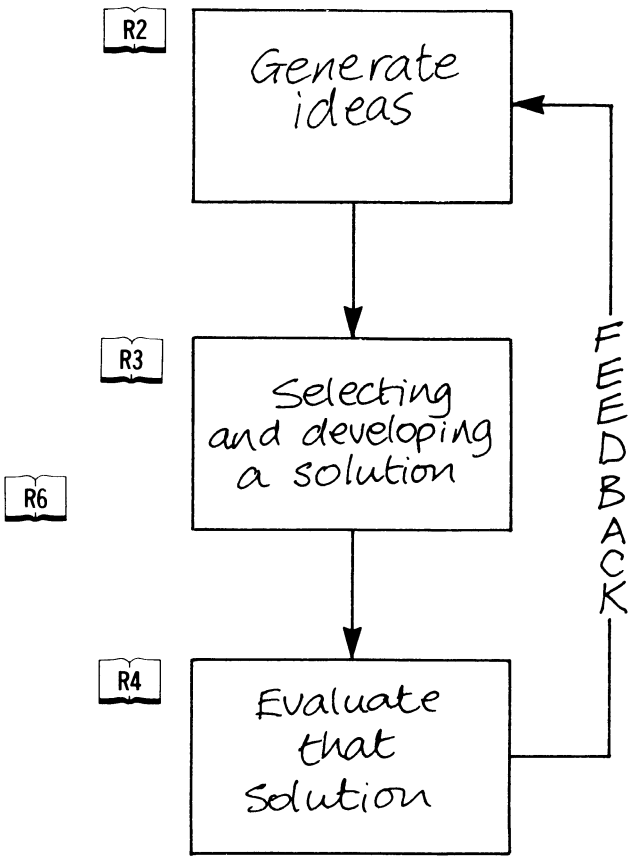
On the large piece of paper, draw a course for your vehicle to go round.

Test your model

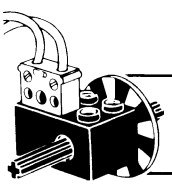
Using the manual controller, test whether your vehicle is capable of going round at least part of the course. if it isn't, you will have to change your design until it can. This is the ***feedback*** part of the design process.

Write a program

Using a Program sheet, write a sequence of instructions which would let somebody else (from another group) make your vehicle follow the course you have drawn.



© This material copyright LEGO Group, 1986. ®LEGO is a registered trade mark belonging to LEGO Group. This material may be freely photocopied *for use in the purchasing institution only.*





Purpose: Assignment 7

To formally introduce a process for solving problems.

Aims

Concept	The problem-solving process.
Skill	Using a model of the process for solving problems.
Skill	Using the information sheets.
Attitude	The willingness and the confidence to solve problems.

Resources (each group)

LEGO 1090 Set
Manual controller
Resources booklet

Resources (Teacher)

Demonstration buggy

Notes

A **teacher-led** activity for groups of three students.

The assignment should be completed in the time allocated.

This assignment is designed as an aide memoire to the problem-solving model. Emphasis should be placed on the process as set out in the Resources booklet, **R1–R5**.

A necessary startpoint is a discussion on the outcome of the previous assignment.

The problem set by the teacher in this assignment will be crucial in order to exemplify the value of the problem-solving process.

For example, the teacher could ask for thoughts on a vehicle to clear rubbish from a sandy beach. Alternatively, Assignment 6 could be explored in greater depth.

Constant reference should be made to the Resources booklet and the results of the previous assignment.

Extension work

Each group reports back to the class.

Troubleshooting

Allow sufficient time to develop the most appropriate solution.

Check for battery and connection problems.

Assignment: 7

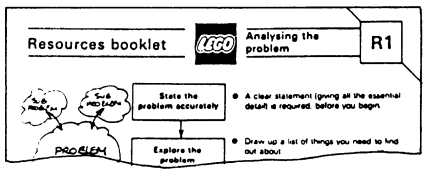
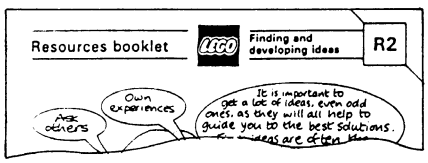
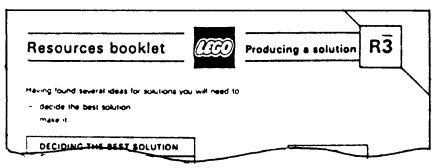
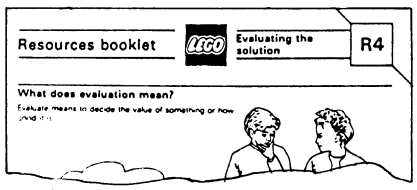
Solving a problem

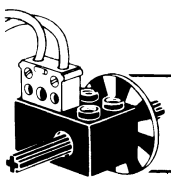
A7

You will need

Resources booklet
Your results from Assignment 6.

This is a model which will help you solve a problem.

Actions	*Outcomes*	*Further explanations in*
Analyse the problem	Criteria list	 **Analysing the problem**
Finding ideas	Several ideas	 **Finding and developing ideas**
Select and develop the best solution	working solution	 **Producing a solution**
Evaluate your solution	test results	 **Evaluating the solution**
Communicate your work	a report	 **Communicating a solution**



a i s e c t

© This material copyright LEGO Group, 1986. ®LEGO is a registered trade mark belonging to LEGO Group. This material may be freely photocopied *for use in the purchasing institution only.*

Assignment: 8

Purpose: Assignment 8

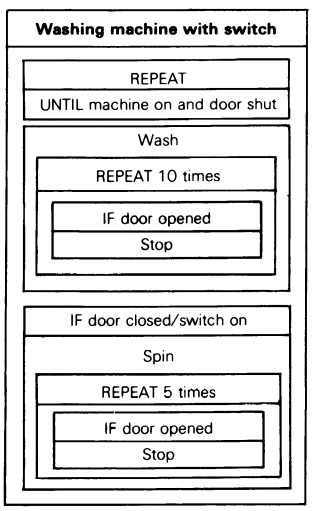
To apply the problem-solving strategy to a software environment and to learn the techniques necessary to produce a structured program.

Aims

Concept	Programming utilises the problems-solving strategy.
Concept	Structured programming results in successful and elegant solutions.
Skill	The ability to apply the problem-solving strategy.
Skill	The ability to use software commands in a LEGO *Lines* programming environment.
Attitude	The willingness and the confidence to solve problems.

Resources (each group)

Computer system
 Computer, monitor and disk drive
Interface with power supply
LEGO *Lines* Program disk
Resources booklet **R1–R5**, **R10–R14**
1090 C Washing machine construction guide
Two light bricks with leads
Two opto-sensor bricks with leads
Program sheets as required



Notes

Study in groups of three, teacher-led/group work.

The assignment should be completed in the time allocated.

The groups will have to build their own washing machine, 1090 C. (If Assignment 7 is not completed in a double lesson it may spill into the first part of an extra session together with the washing machine construction.)

The students will need to refer to **R13**, in particular, and the previous assignments to apply the programming techniques they have used before in Assignments 2–5. They will need to be guided through the development of a program using a ***structured flow diagram*** **R14**. Help for the teacher is found in ***Introduction to the materials*** under ***Program design***.

More than one session is likely to be allocated to this activity.

Extension work

Further problems are set on the Assignment sheet.

Alternatively, groups could develop LEGO *Lines* programs for the vehicle built in Assignments 6 and 7.

Troubleshooting

The accuracy required by a programming environment may cause problems initially, in the precise use of keywords, for example. Students should refer to **R10–R14** to ensure they are using them correctly.

The addition of an emergency switch is an advanced problem. The structure flow diagram opposite provides one solution.

Assignment: 8

Designing a structured program

A8

You will need

A washing machine model 1090 C
Computer system and LEGO *Lines* disk
Resources booklet R1–R5, R10–R14

You are going to

Use the problem-solving process to produce a structured program in order to control a washing machine.

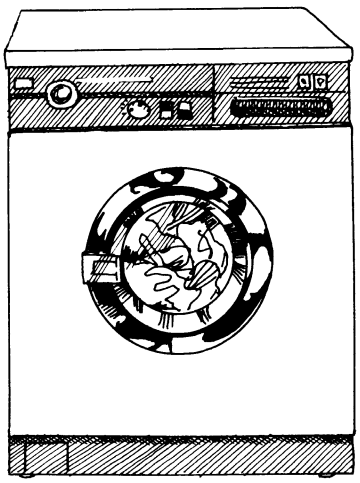
Problem solving

A program is required which turns the motor and the indicator light of a washing machine on when the door is closed and the switch is on.

The washing machine should turn off when the door is opened.

Resources available

The Assignments you have already completed
Resources booklet

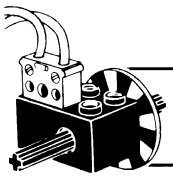


Further problems

Turn the washing machine on if the door is closed and the switch is on. First execute a wash cycle (and light) and then spin (and light).

Add an emergency switch facility which shuts the machine off.

© This material copyright LEGO Group, 1986. ®LEGO is a registered trade mark belonging to LEGO Group. This material may be freely photocopied *for use in the purchasing institution only.*





Purpose: Assignment 9

To provide students with the opportunity to practise problem-solving skills in an open-ended situation.

Aims

Skill	The ability to analyse a problem.
Skill	The ability to search for information.
Skill	The ability to formulate ideas for solutions.
Skill	The ability to evaluate ideas against criteria and select the best.
Skill	The ability to develop ideas into a working model and a software program.
Skill	The ability to evaluate a solution against set criteria.
Skill	The ability to communicate effectively using whatever means is most appropriate.
Attitude	Problem-solving can be enjoyable and successful.
Attitude	Cooperation can lead to more effective problem-solving.

Resources (each group)

1090 LEGO Set
Manual Controller
Computer system
 Computer, monitor and disk drive
Interface with power supply
LEGO ***Lines*** program disk
Resources booklet
Program sheets as required

Notes

Groups of three students (maximum).

There should be as little time constraint as possible.

Each group should be provided with a problem brief from Assignments 9a–9j (or another brief generated from teacher or student discussion).

Assignments 9a–9e use models which may be constructed from the construction guides. 9e is more complex than 9a.

Assignments 9f–9j are adaptations of existing models or require starting from scratch. 9j is the most complex.

Students could be encouraged to prepare a class talk and a report (with a wall display).

Troubleshooting

Students should plan their work using the problem-solving model.

Programs should be tested on paper before attempting to use the computer system.

Check for constructive use of the Resources booklet.

Assignment: 9a

Designing a programmed system

A9a

You will need

1090 LEGO Set
Computer system and LEGO *Lines* disk
Resources booklet
Assignments you have completed

You are going to

Use the problem-solving process.

Sliding Door

A supermarket requires a sliding door system which will allow customers to pass in and out with ease but which remains shut at other times to save heat and energy.

As a development team for *HyperDoor (Electronic Doors) plc* your job is to prepare a report and demonstrate a solution to this problem.



© This material copyright LEGO Group, 1986. ®LEGO is a registered trade mark belonging to LEGO Group. This material may be freely photocopied *for use in the purchasing institution only.*

Assignment: 9b

Purpose: Assignment 9

To provide students with the opportunity to practise problem-solving skills in an open-ended situation.

Aims

Skill	The ability to analyse a problem.
Skill	The ability to search for information.
Skill	The ability to formulate ideas for solutions.
Skill	The ability to evaluate ideas against criteria and select the best.
Skill	The ability to develop ideas into a working model and a software program.
Skill	The ability to evaluate a solution against set criteria.
Skill	The ability to communicate effectively using whatever means is most appropriate.
Attitude	Problem-solving can be enjoyable and successful.
Attitude	Cooperation can lead to more effective problem-solving.

Resources (each group)

1090 LEGO Set
Manual Controller
Computer system
 Computer, monitor and disk drive
Interface with power supply
LEGO ***Lines*** program disk
Resources booklet
Program sheets as required

Notes

Groups of three students (maximum).

There should be as little time constraint as possible.

Each group should be provided with a problem brief from Assignments 9a–9j (or another brief generated from teacher or student discussion).

Assignments 9a–9e use models which may be constructed from the construction guides. 9e is more complex than 9a.

Assignments 9f–9j are adaptations of existing models or require starting from scratch. 9j is the most complex.

Students could be encouraged to prepare a class talk and a report (with a wall display).

Troubleshooting

Students should plan their work using the problem-solving model.

Programs should be tested on paper before attempting to use the computer system.

Check for constructive use of the Resources booklet.

Assignment: 9b

Designing a programmed system

A9b

You will need

1090 LEGO Set
Computer system and LEGO *Lines* disk
Resources booklet
Ideas booklet
Assignments you have completed

You are going to

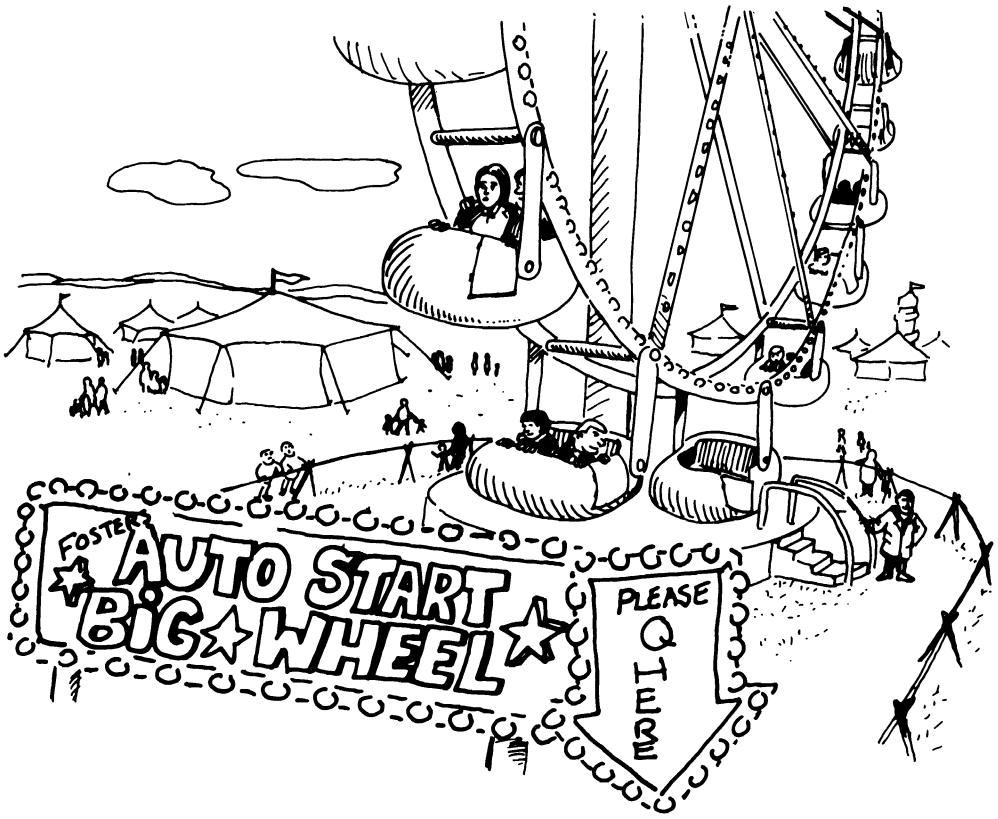
Use the problem-solving process.

Automatic Ferris wheel

A fairground equipment manufacturer has provided your design team with a contract to develop an automatic system for controlling a Ferris wheel ride.

The ride should be safer, more economic and more appealing than the present manual system.

You are required to provide a report and demonstration of a model of your best solution.



© This material copyright LEGO Group, 1986. ®LEGO is a registered trade mark belonging to LEGO Group. This material may be freely photocopied *for use in the purchasing institution only.*

Assignment: 9c

Purpose: Assignment 9

To provide students with the opportunity to practise problem-solving skills in an open-ended situation.

Aims

Skill	The ability to analyse a problem.
Skill	The ability to search for information.
Skill	The ability to formulate ideas for solutions.
Skill	The ability to evaluate ideas against criteria and select the best.
Skill	The ability to develop ideas into a working model and a software program.
Skill	The ability to evaluate a solution against set criteria.
Skill	The ability to communicate effectively using whatever means is most appropriate.
Attitude	Problem-solving can be enjoyable and successful.
Attitude	Cooperation can lead to more effective problem-solving.

Resources (each group)

1090 LEGO Set
Manual Controller
Computer system
 Computer, monitor and disk drive
Interface with power supply
LEGO ***Lines*** program disk
Resources booklet
Program sheets as required

Notes

Groups of three students (maximum).

There should be as little time constraint as possible.

Each group should be provided with a problem brief from Assignments 9a–9j (or another brief generated from teacher or student discussion).

Assignments 9a–9e use models which may be constructed from the construction guides. 9e is more complex than 9a.

Assignments 9f–9j are adaptations of existing models or require starting from scratch. 9j is the most complex.

Students could be encouraged to prepare a class talk and a report (with a wall display).

Troubleshooting

Students should plan their work using the problem-solving model.

Programs should be tested on paper before attempting to use the computer system.

Check for constructive use of the Resources booklet.

Assignment: 9c

Designing a programmed system

A9c

You will need

1090 LEGO Set
Computer system and LEGO *Lines* disk
Resources booklet
Assignments you have completed

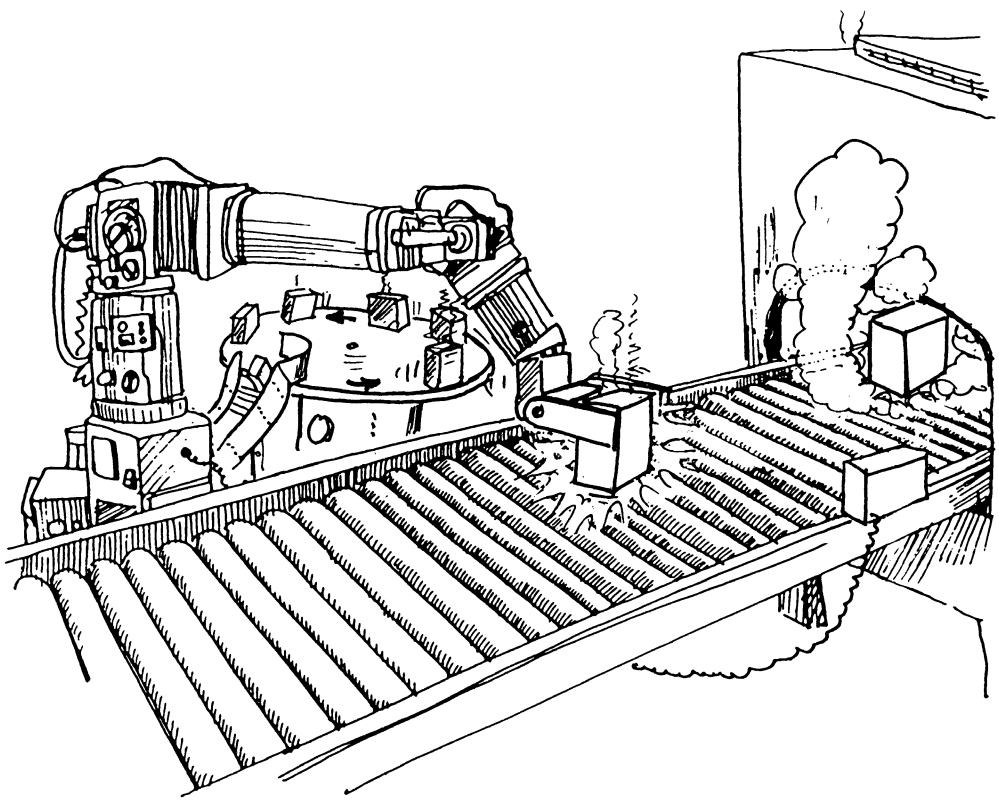
You are going to

Use the problem-solving process.

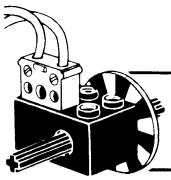
Robot Arm

A manufacturer requires a device which will pick LEGO bricks up from a conveyor belt and place them on a platform situated 10 cms from the conveyor.

Your team must provide a report and a model of this device.



© This material copyright LEGO Group, 1986. ®LEGO is a registered trade mark belonging to LEGO Group. This material may be freely photocopied *for use in the purchasing institution only.*





Purpose: Assignment 9

To provide students with the opportunity to practise problem-solving skills in an open-ended situation.

Aims

Skill	The ability to analyse a problem.
Skill	The ability to search for information.
Skill	The ability to formulate ideas for solutions.
Skill	The ability to evaluate ideas against criteria and select the best.
Skill	The ability to develop ideas into a working model and a software program.
Skill	The ability to evaluate a solution against set criteria.
Skill	The ability to communicate effectively using whatever means is most appropriate.
Attitude	Problem-solving can be enjoyable and successful.
Attitude	Cooperation can lead to more effective problem-solving.

Resources (each group)

1090 LEGO Set
Manual Controller
Computer system
 Computer, monitor and disk drive
Interface with power supply
LEGO ***Lines*** program disk
Resources booklet
Program sheets as required

Notes

Groups of three students (maximum).

There should be as little time constraint as possible.

Each group should be provided with a problem brief from Assignments 9a–9j (or another brief generated from teacher or student discussion).

Assignments 9a–9e use models which may be constructed from the construction guides. 9e is more complex than 9a.

Assignments 9f–9j are adaptations of existing models or require starting from scratch. 9j is the most complex.

Students could be encouraged to prepare a class talk and a report (with a wall display).

Troubleshooting

Students should plan their work using the problem-solving model.

Programs should be tested on paper before attempting to use the computer system.

Check for constructive use of the Resources booklet.

Assignment: 9d

Designing a programmed system

A9d

You will need

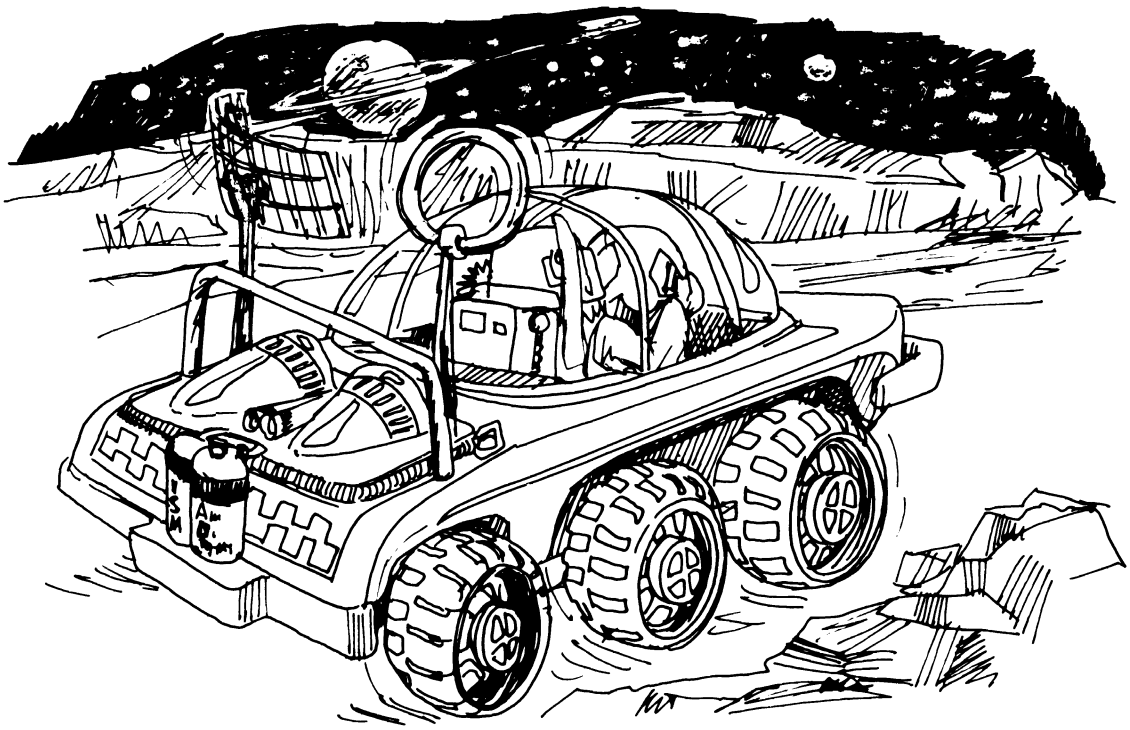
1090 LEGO Set
Computer system and LEGO *Lines* disk
Resources booklet
Assignments you have completed

You are going to

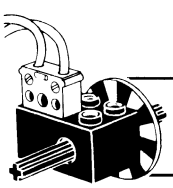
Use the problem-solving process.

Space vehicle

Your development team must prepare a report and build a model of a mobile space vehicle. It should be capable of avoiding obstacles in its path.



© This material copyright LEGO Group, 1986. ®LEGO is a registered trade mark belonging to LEGO Group. This material may be freely photocopied *for use in the purchasing institution only.*





Purpose: Assignment 9

To provide students with the opportunity to practise problem-solving skills in an open-ended situation.

Aims

Skill	The ability to analyse a problem.
Skill	The ability to search for information.
Skill	The ability to formulate ideas for solutions.
Skill	The ability to evaluate ideas against criteria and select the best.
Skill	The ability to develop ideas into a working model and a software program.
Skill	The ability to evaluate a solution against set criteria.
Skill	The ability to communicate effectively using whatever means is most appropriate.
Attitude	Problem-solving can be enjoyable and successful.
Attitude	Cooperation can lead to more effective problem-solving.

Resources (each group)

1090 LEGO Set
Manual Controller
Computer system
 Computer, monitor and disk drive
Interface with power supply
LEGO *Lines* program disk
Resources booklet
Program sheets as required

Notes

Groups of three students (maximum).

There should be as little time constraint as possible.

Each group should be provided with a problem brief from Assignments 9a–9j (or another brief generated from teacher or student discussion).

Assignments 9a–9e use models which may be constructed from the construction guides. 9e is more complex than 9a.

Assignments 9f–9j are adaptations of existing models or require starting from scratch. 9j is the most complex.

Students could be encouraged to prepare a class talk and a report (with a wall display).

Troubleshooting

Students should plan their work using the problem-solving model.

Programs should be tested on paper before attempting to use the computer system.

Check for constructive use of the Resources booklet.

Assignment: 9e

Designing a programmed system

A9e

You will need

1090 LEGO Set
Computer system and LEGO *Lines* disk
Resources booklet
Assignments you have completed

You are going to

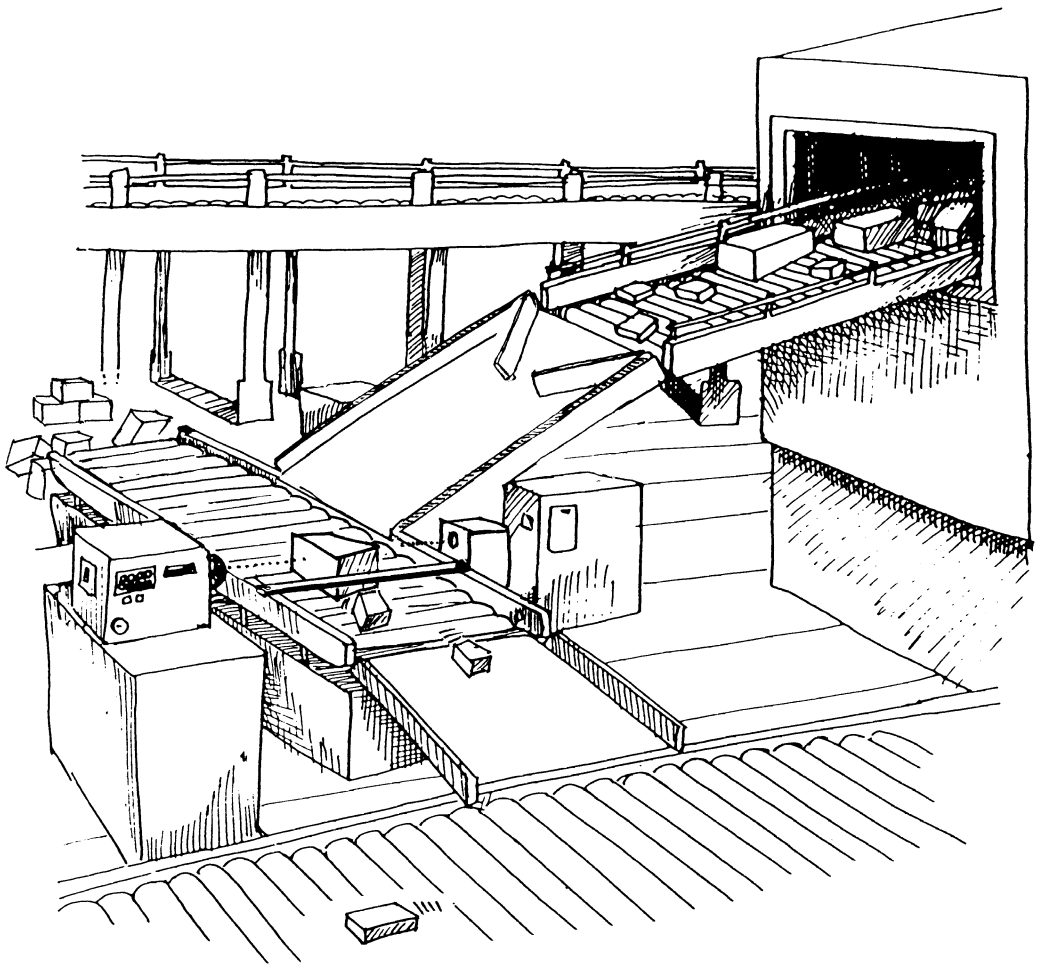
Use the problem-solving process.

Conveyor system

A manufacturer wishes to install a computerised conveyor system in the factory.

The system must be capable of sorting out large boxes on the conveyor from smaller ones.

Your team should build a model of the most appropriate solution, and submit a report on your work to the management.



© This material copyright LEGO Group. 1986. ®LEGO is a registered trade mark belonging to LEGO Group. This material may be freely photocopied *for use in the purchasing institution only.*

Assignment: 9f

Purpose: Assignment 9

To provide students with the opportunity to practise problem-solving skills in an open-ended situation.

Aims

Skill	The ability to analyse a problem.
Skill	The ability to search for information.
Skill	The ability to formulate ideas for solutions.
Skill	The ability to evaluate ideas against criteria and select the best.
Skill	The ability to develop ideas into a working model and a software program.
Skill	The ability to evaluate a solution against set criteria.
Skill	The ability to communicate effectively using whatever means is most appropriate.
Attitude	Problem-solving can be enjoyable and successful.
Attitude	Cooperation can lead to more effective problem-solving.

Resources (each group)

1090 LEGO Set
Manual Controller
Computer system
 Computer, monitor and disk drive
Interface with power supply
LEGO ***Lines*** program disk
Resources booklet
Program sheets as required

Notes

Groups of three students (maximum).

There should be as little time constraint as possible.

Each group should be provided with a problem brief from Assignments 9a–9j (or another brief generated from teacher or student discussion).

Assignments 9a–9e use models which may be constructed from the construction guides. 9e is more complex than 9a.

Assignments 9f–9j are adaptations of existing models or require starting from scratch. 9j is the most complex.

Students could be encouraged to prepare a class talk and a report (with a wall display).

Troubleshooting

Students should plan their work using the problem-solving model.

Programs should be tested on paper before attempting to use the computer system.

Check for constructive use of the Resources booklet.

Assignment: 9f

Designing a programmed system

A9f

You will need

1090 LEGO Set
Computer system and LEGO *Lines* disk
Resources booklet
Assignments you have completed

You are going to

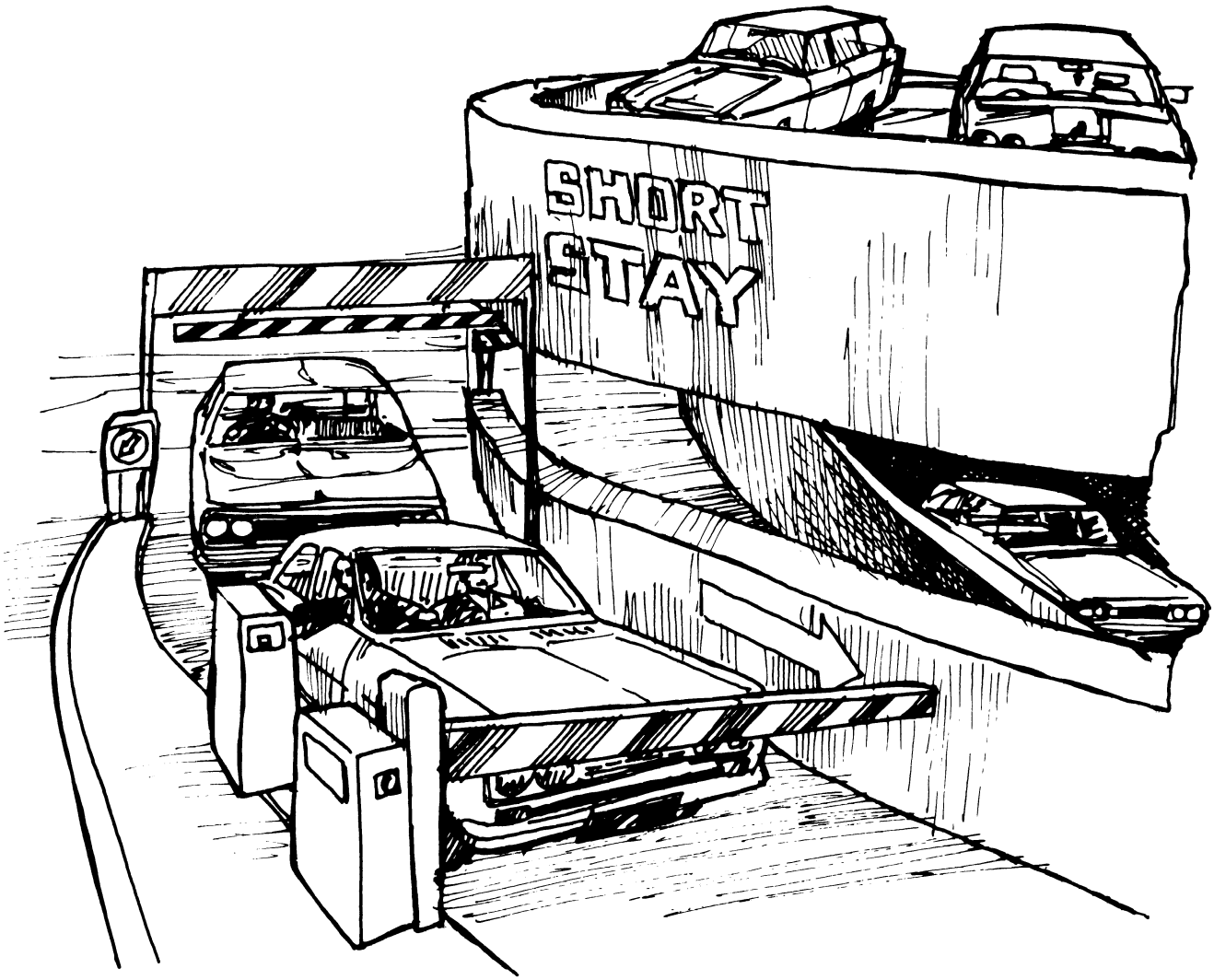
Use the problem-solving process.

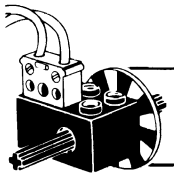
Car Park Barrier

Design an automatic barrier for a multistorey car park.

Care must be taken to ensure that the system your team suggests is efficient and safe.

A model and a report are required.





© This material copyright LEGO Group, 1986. ®LEGO is a registered trade mark belonging to LEGO Group. This material may be freely photocopied *for use in the purchasing institution only.*

Assignment: 9g

Purpose: Assignment 9

To provide students with the opportunity to practise problem-solving skills in an open-ended situation.

Aims

Skill	The ability to analyse a problem.
Skill	The ability to search for information.
Skill	The ability to formulate ideas for solutions.
Skill	The ability to evaluate ideas against criteria and select the best.
Skill	The ability to develop ideas into a working model and a software program.
Skill	The ability to evaluate a solution against set criteria.
Skill	The ability to communicate effectively using whatever means is most appropriate.
Attitude	Problem-solving can be enjoyable and successful.
Attitude	Cooperation can lead to more effective problem-solving.

Resources (each group)

1090 LEGO Set
Manual Controller
Computer system
 Computer, monitor and disk drive
Interface with power supply
LEGO ***Lines*** program disk
Resources booklet
Program sheets as required

Notes

Groups of three students (maximum).

There should be as little time constraint as possible.

Each group should be provided with a problem brief from Assignments 9a–9j (or another brief generated from teacher or student discussion).

Assignments 9a–9e use models which may be constructed from the construction guides. 9e is more complex than 9a.

Assignments 9f–9j are adaptations of existing models or require starting from scratch. 9j is the most complex.

Students could be encouraged to prepare a class talk and a report (with a wall display).

Troubleshooting

Students should plan their work using the problem-solving model.

Programs should be tested on paper before attempting to use the computer system.

Check for constructive use of the Resources booklet.

Assignment: 9g

Designing a programmed system

A9g

You will need

1090 LEGO Set
Computer system and LEGO *Lines* disk
Resources booklet
Assignments you have completed

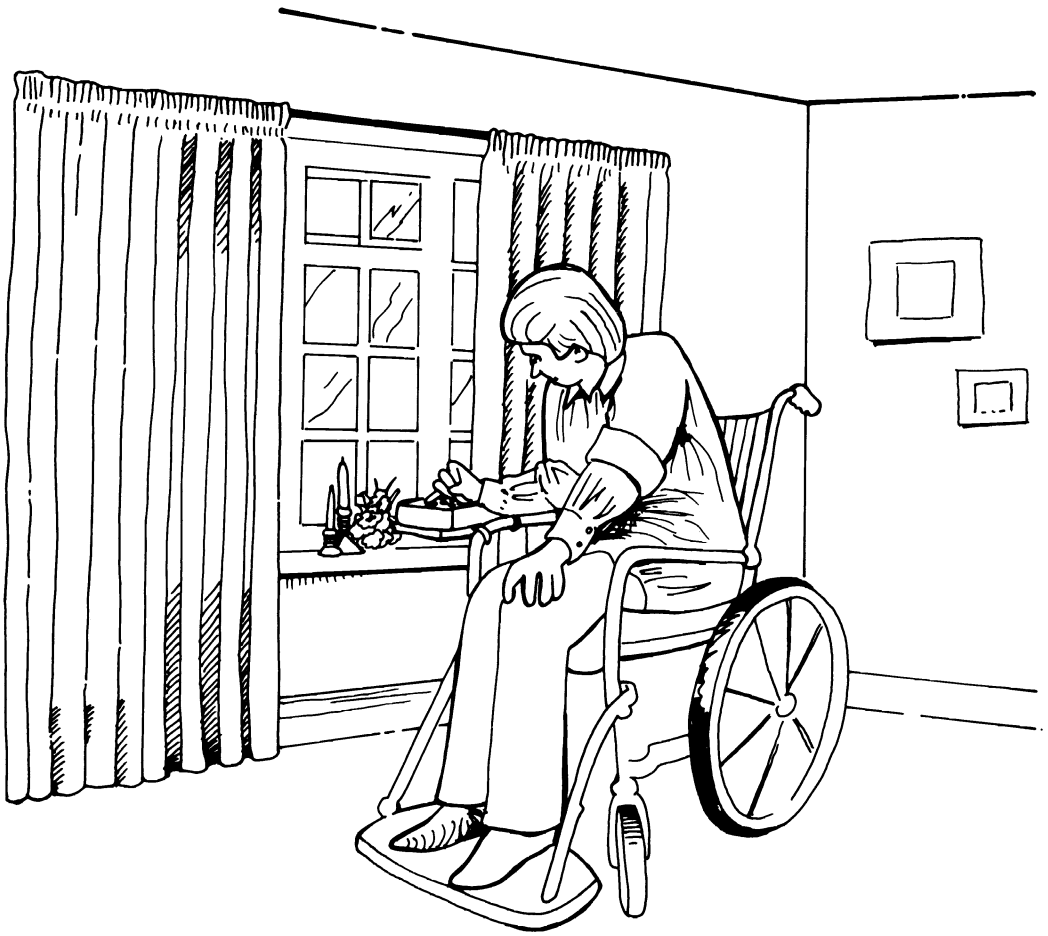
You are going to

Use the problem-solving process.

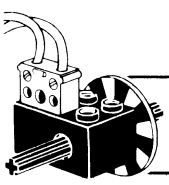
Automatic Curtains

Your team has been commissioned to design a set of curtains which open and close automatically. These would be particularly useful to elderly or disabled people.

Prepare a report of your work and a model of your solution.



© This material copyright LEGO Group, 1986. ®LEGO is a registered trade mark belonging to LEGO Group. This material may be freely photocopied *for use in the purchasing institution only.*





Purpose: Assignment 9

To provide students with the opportunity to practise problem-solving skills in an open-ended situation.

Aims

Skill	The ability to analyse a problem.
Skill	The ability to search for information.
Skill	The ability to formulate ideas for solutions.
Skill	The ability to evaluate ideas against criteria and select the best.
Skill	The ability to develop ideas into a working model and a software program.
Skill	The ability to evaluate a solution against set criteria.
Skill	The ability to communicate effectively using whatever means is most appropriate.
Attitude	Problem-solving can be enjoyable and successful.
Attitude	Cooperation can lead to more effective problem-solving.

Resources (each group)

1090 LEGO Set
Manual Controller
Computer system
 Computer, monitor and disk drive
Interface with power supply
LEGO ***Lines*** program disk
Resources booklet
Program sheets as required

Notes

Groups of three students (maximum).

There should be as little time constraint as possible.

Each group should be provided with a problem brief from Assignments 9a–9j (or another brief generated from teacher or student discussion).

Assignments 9a–9e use models which may be constructed from the construction guides. 9e is more complex than 9a.

Assignments 9f–9j are adaptations of existing models or require starting from scratch. 9j is the most complex.

Students could be encouraged to prepare a class talk and a report (with a wall display).

Troubleshooting

Students should plan their work using the problem-solving model.

Programs should be tested on paper before attempting to use the computer system.

Check for constructive use of the Resources booklet.

Assignment: 9h

Designing a programmed system

A9h

You will need

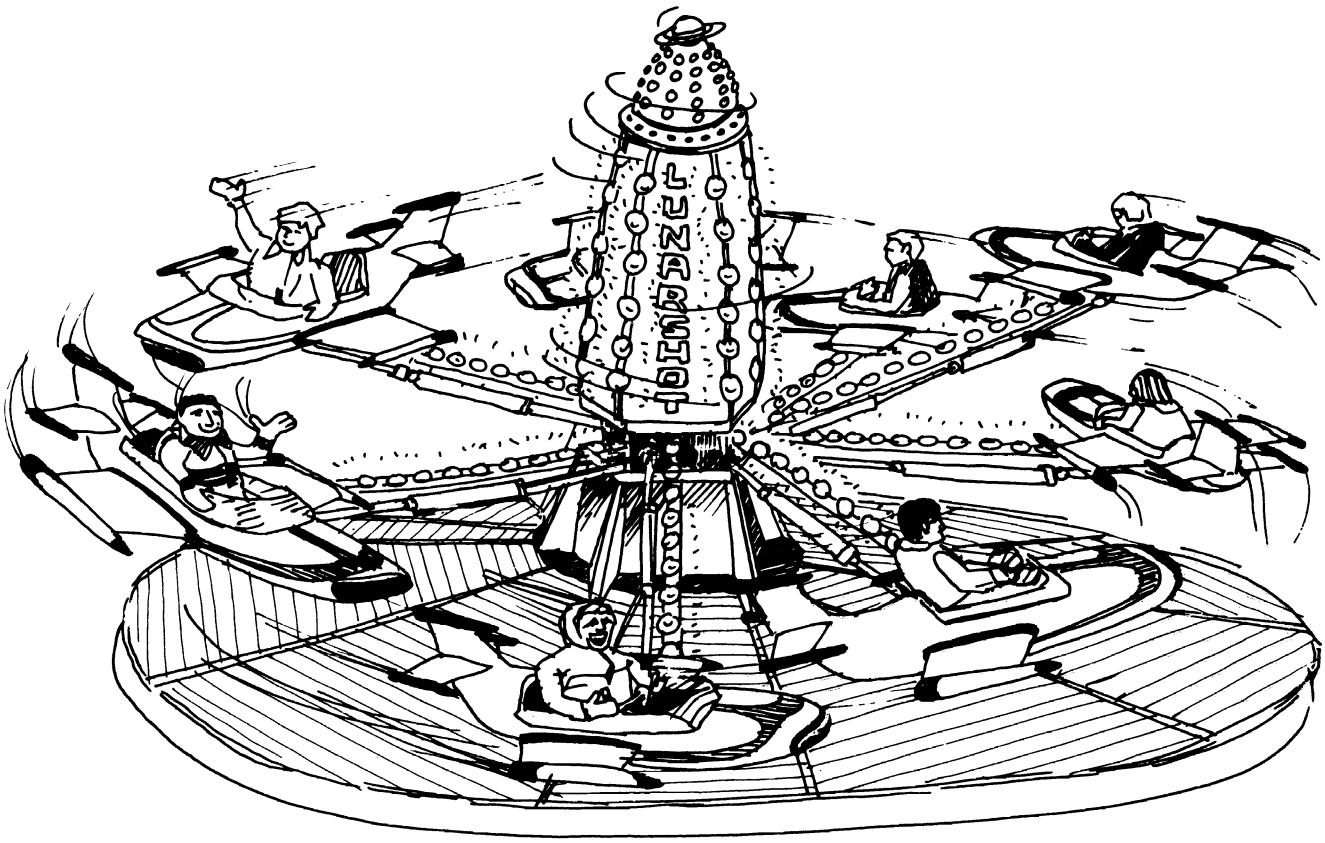
1090 LEGO Set
Computer system and LEGO *Lines* disk
Resources booklet
Assignments you have completed

You are going to

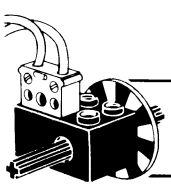
Use the problem-solving process.

Roundabout

A leisure park wishes to install an automatically controlled roundabout. They have commissioned your team to produce a working model and a report of your design.



© This material copyright LEGO Group, 1986. ®LEGO is a registered trade mark belonging to LEGO Group. This material may be freely photocopied *for use in the purchasing institution only.*



Assignment: 9i

Purpose: Assignment 9

To provide students with the opportunity to practise problem-solving skills in an open-ended situation.

Aims

Skill	The ability to analyse a problem.
Skill	The ability to search for information.
Skill	The ability to formulate ideas for solutions.
Skill	The ability to evaluate ideas against criteria and select the best.
Skill	The ability to develop ideas into a working model and a software program.
Skill	The ability to evaluate a solution against set criteria.
Skill	The ability to communicate effectively using whatever means is most appropriate.
Attitude	Problem-solving can be enjoyable and successful.
Attitude	Cooperation can lead to more effective problem-solving.

Resources (each group)

1090 LEGO Set
Manual Controller
Computer system
 Computer, monitor and disk drive
Interface with power supply
LEGO ***Lines*** program disk
Resources booklet
Program sheets as required

Notes

Groups of three students (maximum).

There should be as little time constraint as possible.

Each group should be provided with a problem brief from Assignments 9a–9j (or another brief generated from teacher or student discussion).

Assignments 9a–9e use models which may be constructed from the construction guides. 9e is more complex than 9a.

Assignments 9f–9j are adaptations of existing models or require starting from scratch. 9j is the most complex.

Students could be encouraged to prepare a class talk and a report (with a wall display).

Troubleshooting

Students should plan their work using the problem-solving model.

Programs should be tested on paper before attempting to use the computer system.

Check for constructive use of the Resources booklet.

Assignment: 9i

Designing a programmed system

A9i

You will need

1090 LEGO Set
Computer system and LEGO *Lines* disk
Resources booklet
Assignments you have completed
Plasticine block

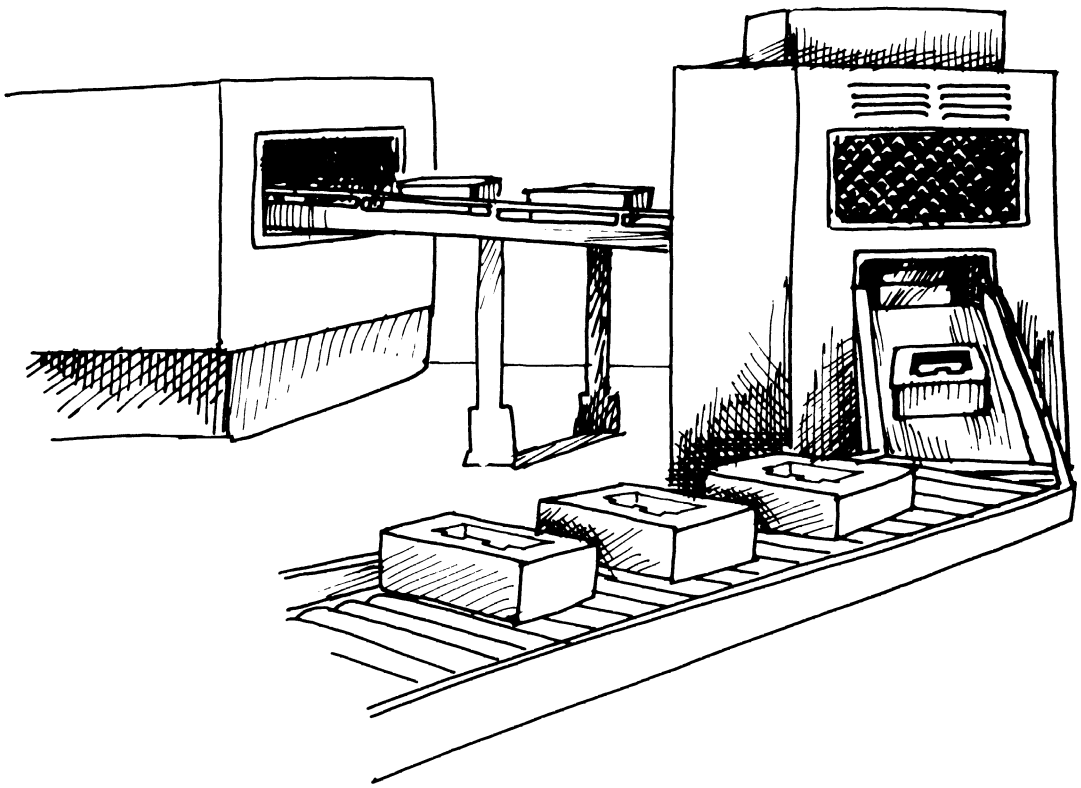
You are going to

Use the problem-solving process.

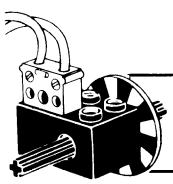
Pressing machine

Your team should design a device which will automatically stamp the pattern of a LEGO brick onto a block of plasticine when it is placed onto a platform.

A report and a working model are required.



© This material copyright LEGO Group, 1986. ®LEGO is a registered trade mark belonging to LEGO Group. This material may be freely photocopied *for use in the purchasing institution only.*



Assignment: 9j

Purpose: Assignment 9

To provide students with the opportunity to practise problem-solving skills in an open-ended situation.

Aims

Skill	The ability to analyse a problem.
Skill	The ability to search for information.
Skill	The ability to formulate ideas for solutions.
Skill	The ability to evaluate ideas against criteria and select the best.
Skill	The ability to develop ideas into a working model and a software program.
Skill	The ability to evaluate a solution against set criteria.
Skill	The ability to communicate effectively using whatever means is most appropriate.
Attitude	Problem-solving can be enjoyable and successful.
Attitude	Cooperation can lead to more effective problem-solving.

Resources (each group)

1090 LEGO Set
Manual Controller
Computer system
 Computer, monitor and disk drive
Interface with power supply
LEGO ***Lines*** program disk
Resources booklet
Program sheets as required

Notes

Groups of three students (maximum).

There should be as little time constraint as possible.

Each group should be provided with a problem brief from Assignments 9a–9j (or another brief generated from teacher or student discussion).

Assignments 9a–9e use models which may be constructed from the construction guides. 9e is more complex than 9a.

Assignments 9f–9j are adaptations of existing models or require starting from scratch. 9j is the most complex.

Students could be encouraged to prepare a class talk and a report (with a wall display).

Troubleshooting

Students should plan their work using the problem-solving model.

Programs should be tested on paper before attempting to use the computer system.

Check for constructive use of the Resources booklet.

Assignment: 9j

Designing a programmed system

A9j

You will need

1090 LEGO Set
Computer system and LEGO *Lines* disk
Resources booklet
Assignments you have completed

You are going to

Use the problem-solving process.

Stage Effect

Your team has been commissioned to design a head to be mounted on a wall as part of a fantasy play. The head is required to 'come alive' each time one of the cast passes a certain spot.

Prepare a report of your ideas and a working model.



© This material copyright LEGO Group, 1986. ®LEGO is a registered trade mark belonging to LEGO Group. This material may be freely photocopied *for use in the purchasing institution only.*



Section 3

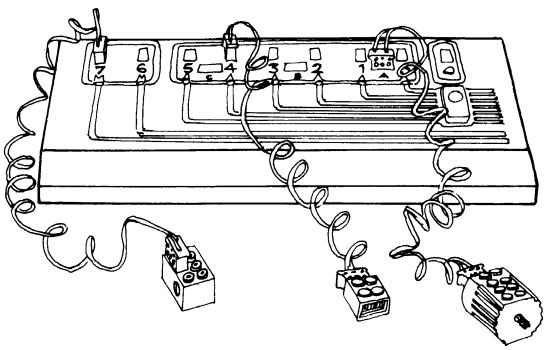
Resources booklet

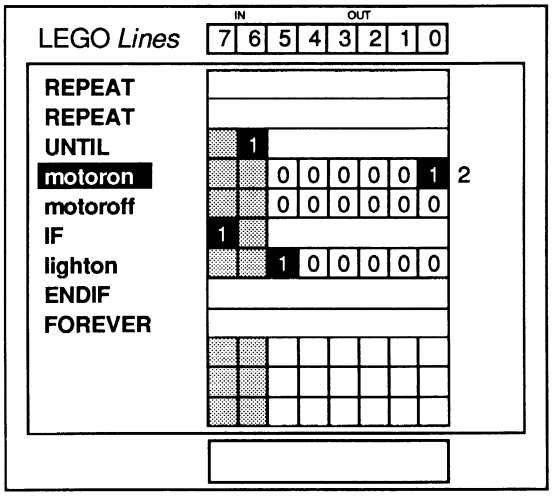
Resources booklet

Resources booklet

This booklet is designed to give you useful information and to help you as you work your way through the Programmable Systems material.

It also provides you with a great deal of information about Technology and you should be able to use it as a resource at any time.





Designed and produced by:

Tecmedia Limited
Loughborough
Leicestershire
England
Tel 0509 230248

ISBN 1 7316 0550 0 Programmable Systems (Whole Pack)
ISBN 1 7316 0721 X Programmable Systems (Resources Booklet)

® LEGO is a registered trade mark
© 1987 LEGO Group

These materials, both text and computer software, are fully protected by international copyright. If you make a copy and allow anyone else to retain it (***even if you give it to them***), you are committing an offence and are liable to prosecution.

You are advised, however, that the software and many of the materials are intended as photocopy masters and may be freely copied ***for use in the purchasing institution***.

Guide to contents

Page

Organising yourself to solve problems

R1 **Analysing the problem** **3.5**
stating the problem — exploring the problem — developing an understanding of the problem — developing criteria and criteria tests

R2 **Finding and developing ideas** **3.6**
brainstorming — using mindmaps — techniques for developing ideas

R3 **Producing a solution** **3.7**
the results of criteria tests — the best or most appropriate solution — the best use of time

R4 **Evaluating the solution** **3.8**
meaning of evaluation — making decisions — analysing the results of tests

R5 **Communicating your solution** **3.9**
written reports — spoken reports — sketches and diagrams

Using the equipment

R6 **Manual controller** **3.10**
connecting models to the controller — keystrip

R7 **Computer system** **3.11**
hardware items — connecting the system together — connecting power to the system — Apple IIe connections

R8 **The LEGO interface — output** **3.12**
light brick connections — motor connections — using output connections — open loop control

R9 **The LEGO interface — input** **3.13**
using the input connections — the opto-sensor brick — closed loop control

R10 **LEGO *Lines*** **3.14**
mouse and keyboard conventions — loading and saving programs — writing programs (keys) — layout of the screen — layout of a line — changing lines — LEGO ***Lines*** commands

R11 **LEGO *Lines* User Guide** **3.15**
explanation of a line — using the interface indicator lights — sample programs — error codes

Designing control programs using the LEGO materials

Page

R12 Feedback **3.20**
importance of feedback — examples of feedback

R13 Developing a program **3.21**
using the technological process to design programs — analysis, ideas, solutions, building, testing and communicating

R14 Structured flow diagrams (*sfd*'s) **3.22**
sequence of instructions — actions — tests and loops

R15 Vehicles — ideas on structures **3.24**

R16 Vehicles — ideas on drive mechanisms **3.25**

R17 Vehicles — ideas on steering mechanisms **3.26**

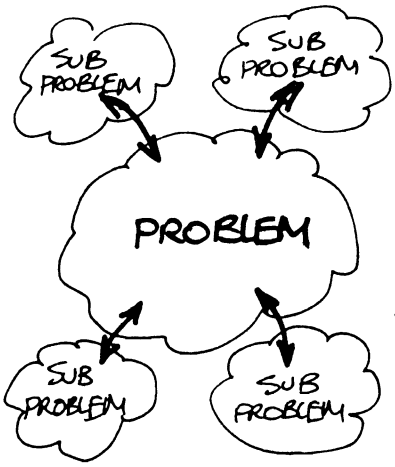
R18 Vehicles — ideas on gearboxes **3.27**

R19 Ideas — indoors **3.28**

R20 Ideas — outdoors **3.29**

R21 Ideas — at work **3.30**

R22 Glossary of terms **3.31**



State the problem accurately

- A clear statement (giving all the essential detail) is required, before you begin.

Explore the problem

- Draw up a list of things you need to find out about.
- Decide how you are going to find out about the things on your list.

Develop a deeper understanding

- Carry out investigations to find the answers to the (sub) problems you can identify.

Develop criteria

- Discuss or make notes about what the solution should do.
- Develop tests against which you will judge your solution.

Criteria

A list of statements describing what is required of the solution.

How do I make a list of criteria?

Here are some ways

Ask yourself



See what others have done



Ask others





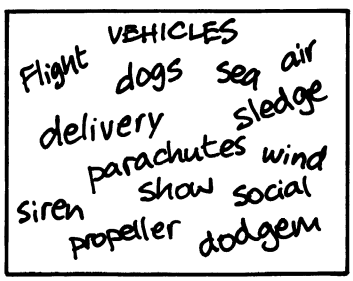
Working in groups to develop ideas

Brainstorming

Get ideas down quickly
Everybody can say something
Any idea will do
Only one or two words per idea

Hints

Give important words plenty of space
Group words together in topic areas

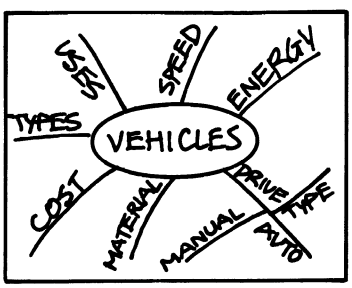


Mindmaps

Build ideas in logical patterns
Provides a good record
Promotes clear thinking
Ideas can be used at the end

Hints

Use words like ***how***, ***what***, ***when***, ***why***, ***where***
Print on the lines of mindmaps
Use words which carry **meaning**
Branch in any direction and in any order on map
Link ideas using lines and arrows



Having found several ideas for solutions you will need to:

— decide the best solution

— make it.

DECIDING THE BEST SOLUTION

ANALYSE THE SOLUTIONS

Decide how well each solution will perform the criteria tests

DECIDE THE BEST SOLUTION

Decide which solution comes out overall best
or
Make up a new solution from the best parts of the others.

You will need

- Solution ideas
- Criteria tests

Making best use of time

There are a number of ways of analysing the solutions. You might be tempted to think that building and testing each solution would be the best way but this takes a lot of time. You will need to take short cuts by using your experience and the experience of others in your group to reach decisions.



- Test each solution
- Some practical investigations with experience
- Your own experience and the experiences of others
- Guess

What does evaluation mean?

Evaluate means to decide the value of something or how good it is.



When you evaluate the final solution you should

- Carry out the criteria tests
- Analyse the test results

When you are doing the tests

Decide on

- The success of each test
- Reasons for weaknesses
- Possible modifications

Note

There could be a number of reasons for weaknesses:

- —Poor design
- —Limitations of kit
- —Lack of time
- —Poor criteria

When you are analysing the results

Think about

- A general description of how well your solution solves the problem
- Possible improvements
- How easily the model solution could be adapted for real use

Purpose

Give useful information to others
Convince others your ideas are best
Record ideas and information

Your report should be

Easily understood **Quickly understood** **Interesting**

WRITTEN REPORT

Contents

Title *Author(s)*

The problem — short statement (a few lines)

The solution — short statement (a few lines)

Criteria used

Solutions considered (with evaluation of each)

Reasons for choosing your solution

Development (problems encountered and how they were overcome)

Results of evaluating the final solution

SPOKEN REPORT

Preparation

Good large diagrams
(only essential detail)

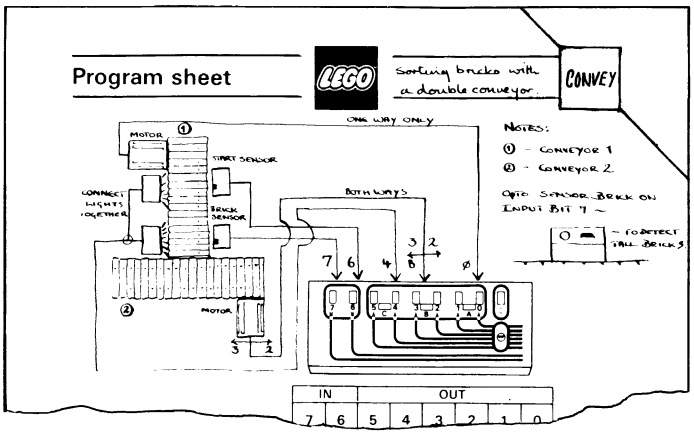
Content of talk

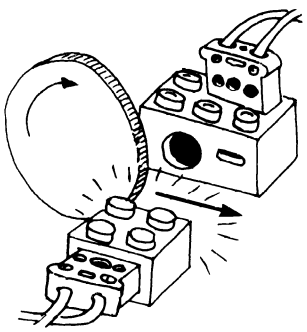
The problem
The criteria used
Solutions considered
The solution chosen
The reasons for choice
Development problems

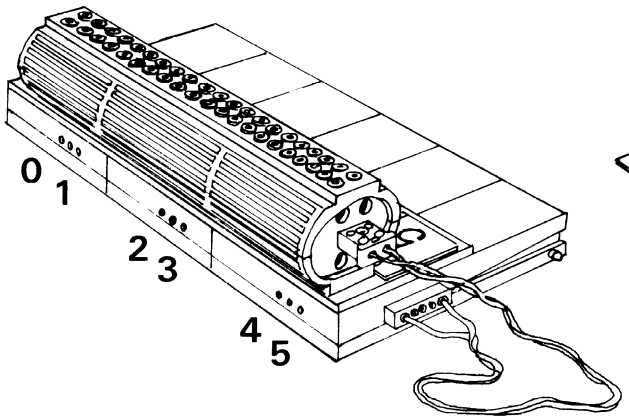
Hints

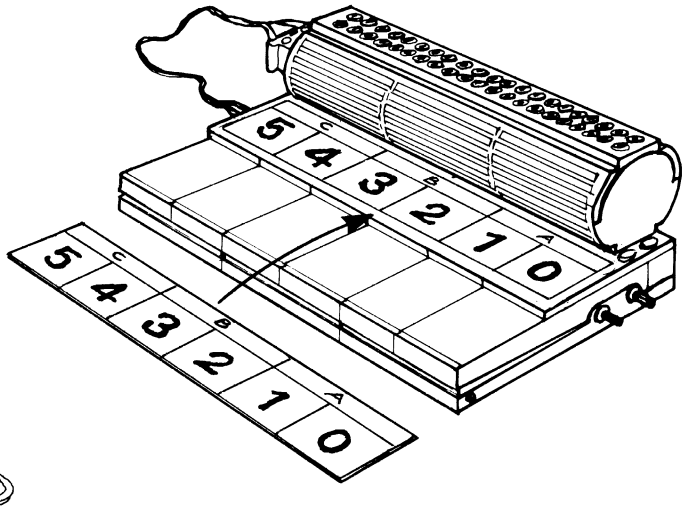
Keep the task simple
Talk to the whole audience
Do not speak too fast
Speak loud enough to be heard

Remember: sketches and diagrams are worth many words.



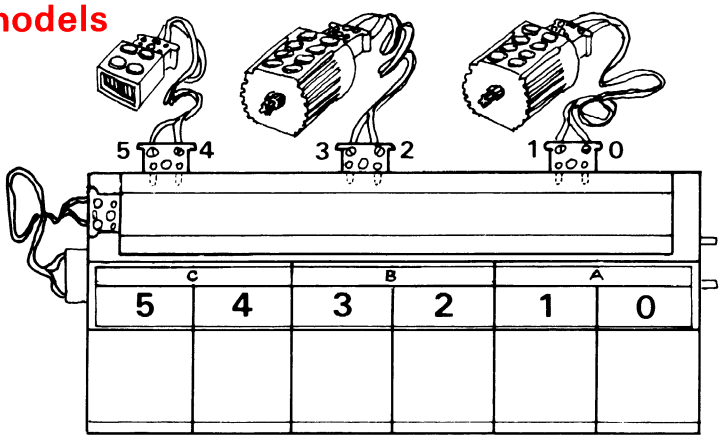




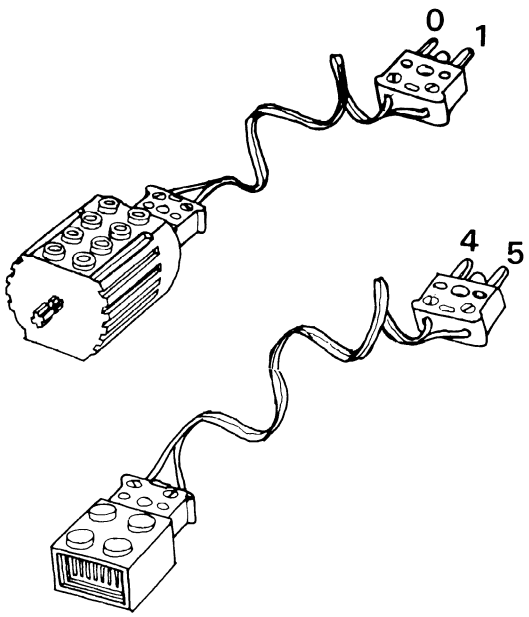


Put keyboard strip over keys

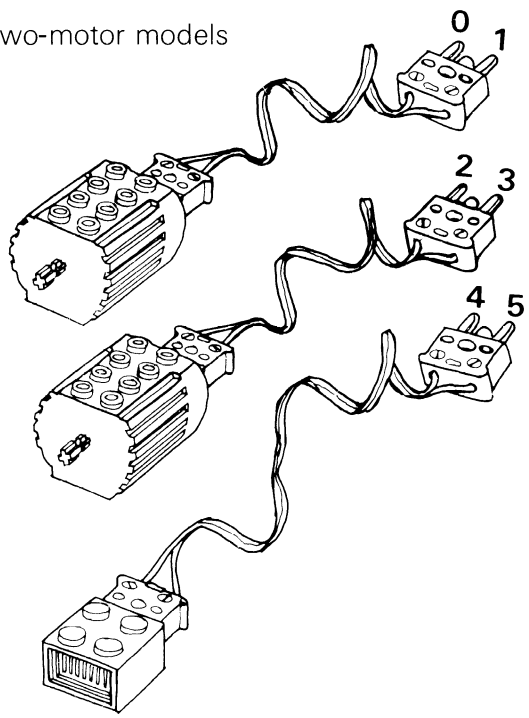
Connecting models



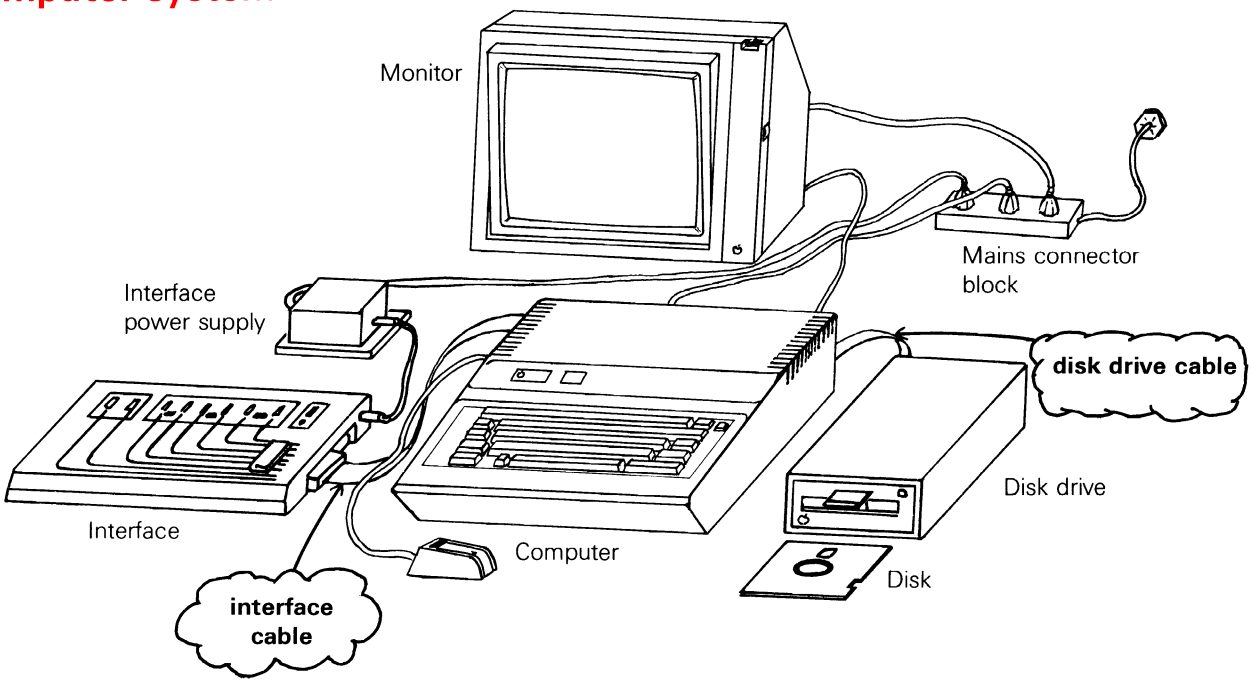
One-motor models



Two-motor models



Computer system

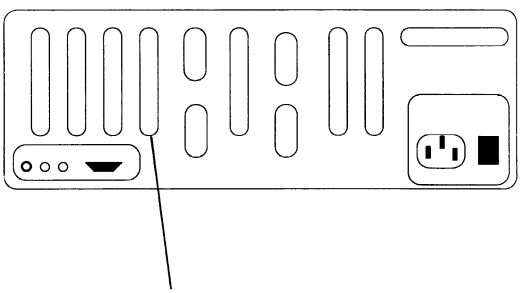


Apple® IIe connections

Installing a LEGO Slot Card

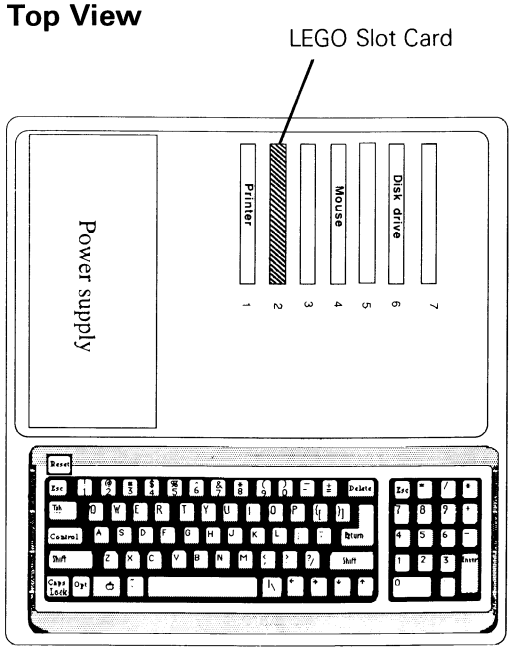
For full details on how to install your LEGO Slot Card, please refer to the *Technical reference* section, page 4.1.

Rear View



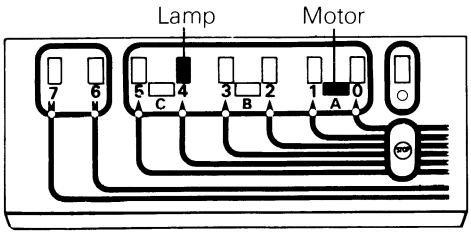
The ribbon cable from the LEGO Interface box plugs in here.

Top View



The LEGO Slot Card also fits into slot 2 of the Apple® IIGS computer.

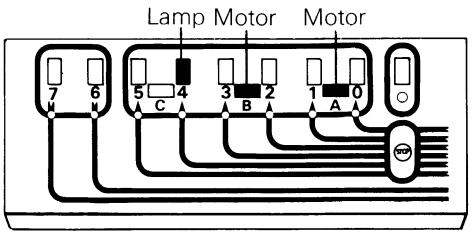
Interface connections for One motor models



4 v power supply

STOP — Press to switch off all outputs

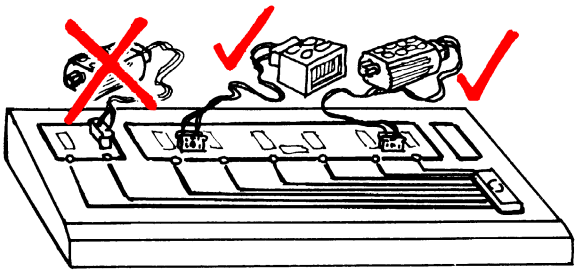
Interface connections for Two motor models

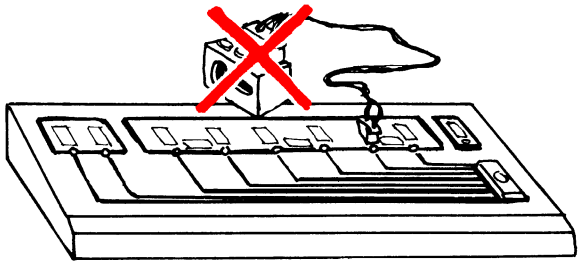


4 v power supply

STOP — Press to switch off all outputs

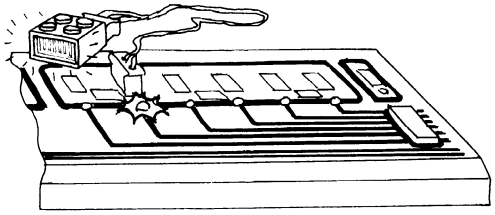
Output connections



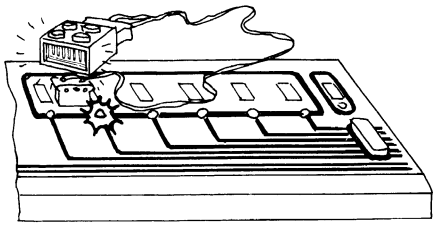


Use of output connections

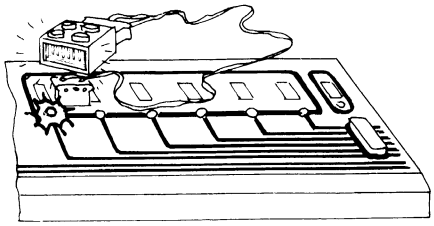
Light on



Light on

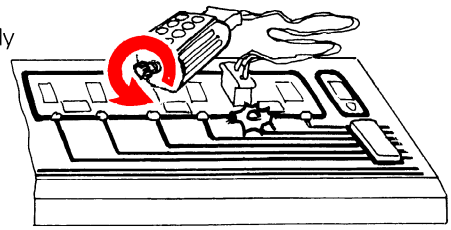


Light on



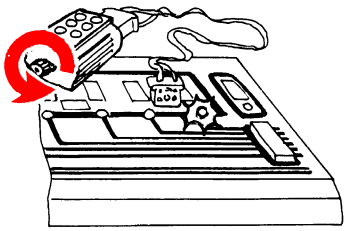
Motor on

One way only

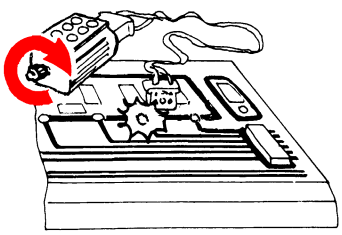


Motor on

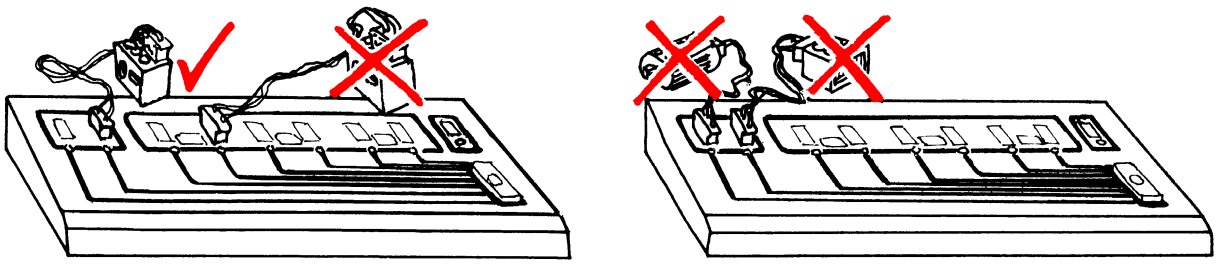
One way



Other way



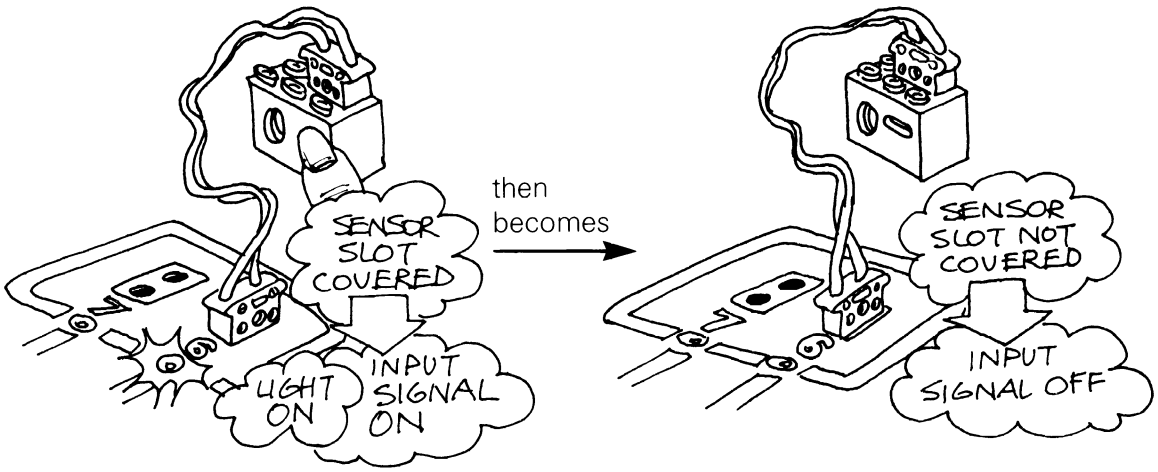
Input connections



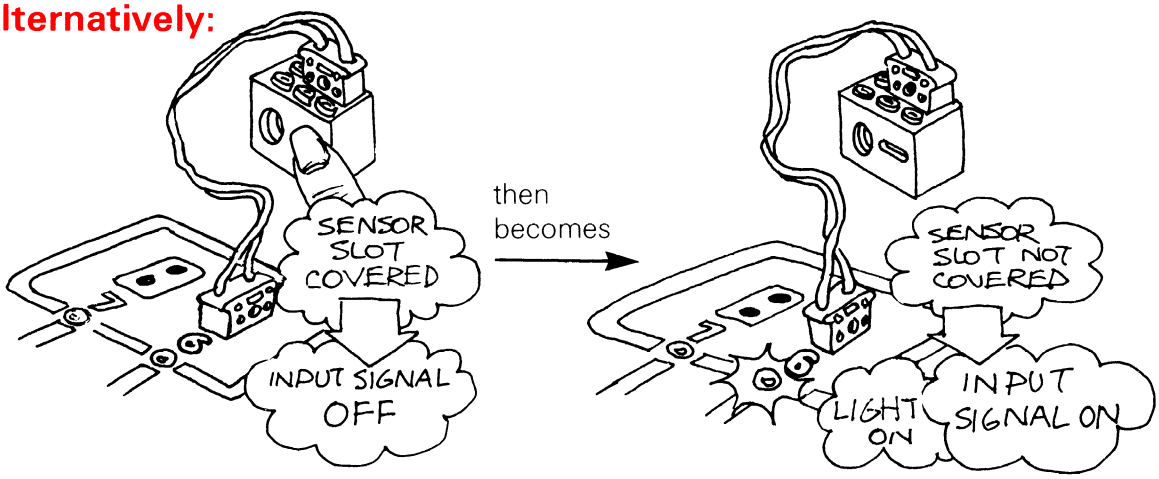
Opto-sensor brick

Electronic sensors take information from the outside world and produce electronic signals to represent that information.

The opto-sensor itself nestles in a hole in the brick. It will react to both external light and to reflected light. Care must be taken in that it will sometimes be on initially, other times off initially. You can force the sensor to be either on or off by shining a light into it from the light brick.



Alternatively:



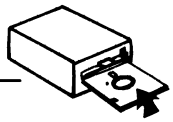
Connect the opto-sensor brick to bit 6 on the interface. Is the indicator light on or off? When you place your finger over the sensor it will respond to ***changes*** of light conditions.

Change the state of bit 6 a number of times.

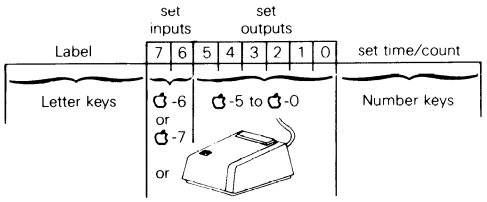
Quick Reference Guide

Loading LEGO *Lines*

1 Connect the computer system . . .
2 Insert LEGO ***Lines*** disk . . .
3 Turn the computer on.



Writing a program

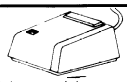


Keyboard convention



PRESS	Press key and release
HOLD	Hold key down
Apple-key or **Apple-Key**	Hold open Apple key down . . . Press second key and release . . . Release open Apple key.
SELECT	Use arrow keys to move cursor to desired position and press **RETURN**.

Mouse convention



Move pointer to this position, click once.

Move pointer to this position, click twice.

	1 **MOVE ...** the cursor	2 **TEST ...** a line	3 **RUN ...** a program from the beginning	4 **RUN ...** part of a program	5 **DELETE ...** a character	6 **DELETE ...** a line
Keyboard	Arrow keys Apple- ↑ to top line Apple- ↓ to bottom line	Apple-T	Apple-R	Position cursor on first line of portion to be run . . . Press **ESCAPE**. Select **RUN** Select **PART** Select portion to be run. Press **RETURN**	Press **DELETE**	Apple-D
Mouse	move then click	TEST hold button down	RUN	Position cursor on first line of portion to be run . . . RUN PART Select portion to be run. Press **RETURN**.	—	—

	7 **BLANK ...** out a line	8 **INSERT ...** a line	9 **STOP ...** a program that is running	10 **PAUSE ...** during a program	11 **SAVE ...** a program	12 **LOAD ...** a program
Keyboard	Apple-B	Apple-I	Press **ESCAPE**. (Apple-R to run again from start)	Press **SPACE** bar to pause Press **SPACE** bar to continue **ESC** to stop	Apple-S	Apple-L
Mouse	—	—	—	—	DISK SAVE	DISK LOAD

	13 **VIEW ...** structure of program	14 **PRINT ...** the program	15 **ERASE ...** a program from memory	16 **DELETE ...** a program from disk	17 **CONTENTS ...** of the disk	18 **HELP ...** with key functions
Keyboard	Apple-V	Apple-P	Press **ESCAPE** Select **MISC** Select **NEW**	Press **ESCAPE** Select **DISK** Select **DELETE**	Press **ESCAPE** Select **DISK** Select **CONTENTS**	Apple-?
Mouse	VIEW	PRINT	MISC NEW	DISK DELETE	DISK CONTENTS	INFO HELP

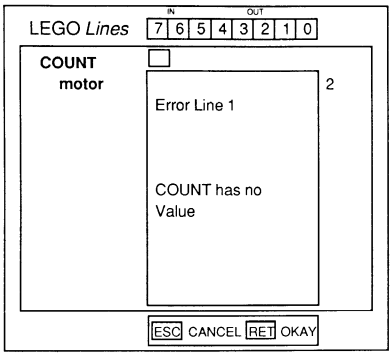
Error messages

Before running or viewing a program, ***Lines*** does various checks to ensure the structures within the program are correct. If there are any errors then a message will be displayed in the form:

ERROR LINE (line number)

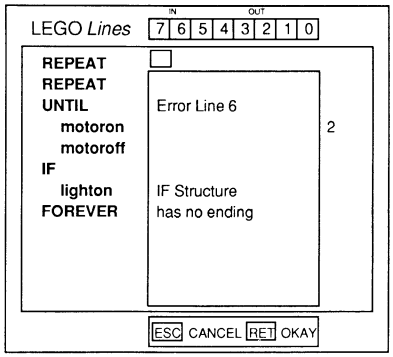
Below this message the error messages appear as follows:

1 **Structures too deeply nested**
2 **COUNT has no value**
3 **If structure has no ending**
4 **Repeat structure has no ending**
5 **If structure has no beginning**
6 **Repeat structure has no beginning**



Explanations of error messages

1 You have more than eight levels of nexting in the program structure.
2 You need a numerical value in the line containing **COUNT**.
3 Either there is an **ENDIF** message or the **IF-ENDIF** structure is broken.
4 Either there is an **ENDREPEAT/UNTIL/FOREVER** missing *or* **REPEAT** has a wrong ending, *or* the **REPEAT** structure has been broken.
5 **ENDIF** without **IF** or the **IF** structure is broken.
6 **ENDREPEAT**, **UNTIL**, **FOREVER** have no **REPEAT**, *or* start the wrong way *or* the structure has been broken.



After each message is displayed you may press **RETURN** to continue the check. Pressing **ESCAPE** will return you to the edit screen.

A program with errors will not run!

Disk related errors

The possible disk errors are:

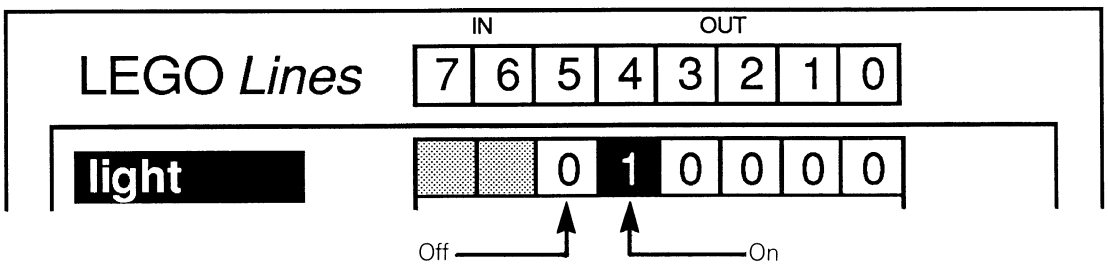
Disk full
Bad disk (includes unformatted disk)
Disk Write-Protected

The LEGO models can be controlled by the LEGO ***Lines*** program running on the microcomputer.

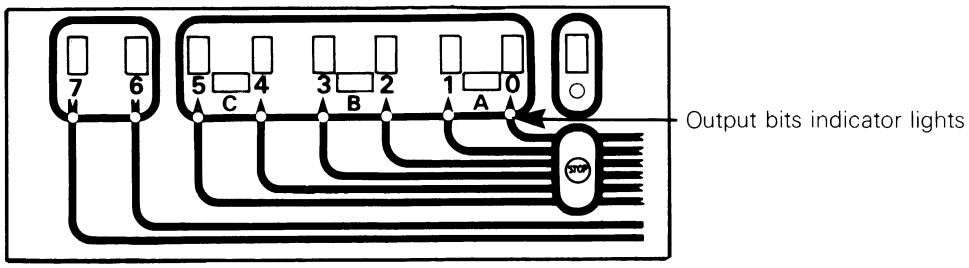
Physical control over the models is achieved by using a pattern of signals, made up of 8 bits, which link the model and the computer together via the interface.

Each of the 8 bits may be set on or off and one pattern of the bits controls the model in a different way to another pattern.

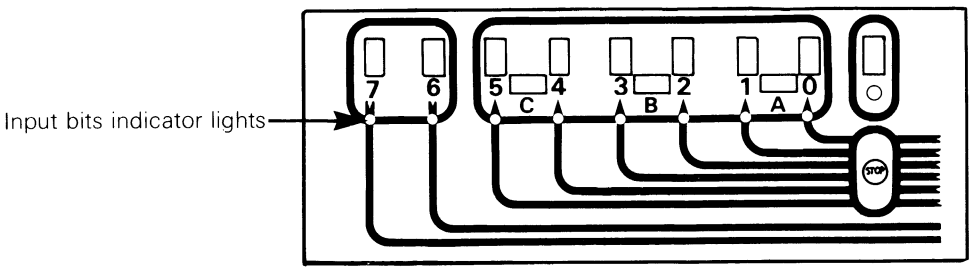
On the screen, a bit which is set on is represented by a figure one in that bit's box whereas a bit which is set off is represented by a zero.



The LEGO ***Lines*** program is designed to use six of the eight bits as **output** bits. These control the motor and lamp devices from the LEGO set. These are numbered 0-5 on the interface and the indicator lights will show whether they are set on or off at any time.



The other two bits are **input** bits. These are designed to receive messages from the LEGO opto-sensors. These are the bits numbered 6 and 7, and the indicator lights will, again, show their on or off state.

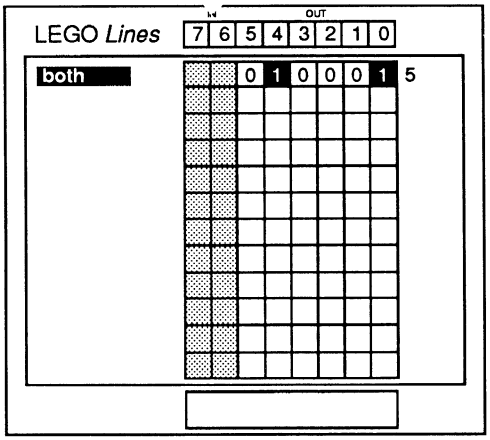


LEGO ***Lines*** makes use of these bits to physically control the LEGO devices.

Logical control over the models is achieved by using the commands and statements available in LEGO ***Lines*** called ***reserved words***.

Using the commands available and combining them in certain ways, the user can build up a sequence of lines which make up a control program.

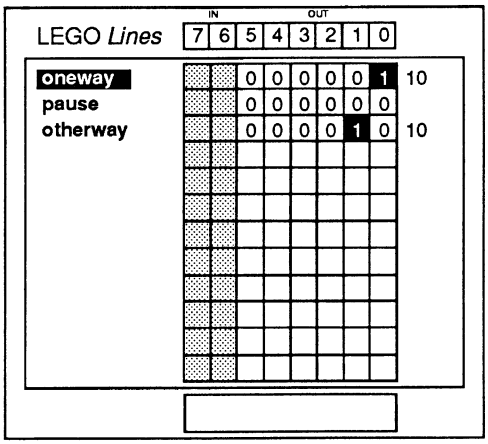
The smallest program which can be created is a single line which will instruct LEGO ***Lines*** to control the devices attached to the computer system in a particular way. Each line can be given a label to indicate what is happening to the model when that line is being executed.



Program "TWO"

This program lights a lamp and turns on a motor.

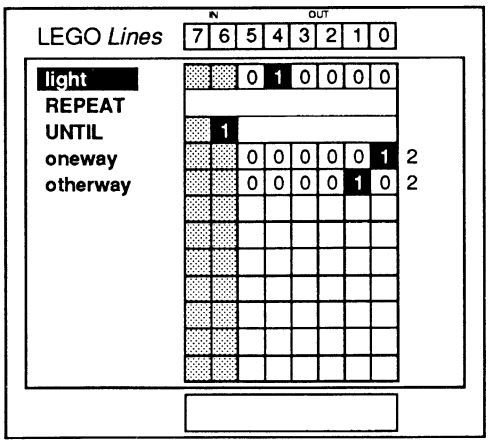
A program usually contains a sequence of lines and these will be executed in the order they are set out from top to bottom.



Program "FOUR"

This program makes a motor revolve one way and then the other way.

You will notice that these two programs do not involve any lines which have input bits in them.



Program "MLB"

The input bits allow LEGO ***Lines*** to make decisions and, when required, to give commands to control a model.

This program lights a lamp and then waits until input bit 6 is turned on before it makes the motor rotate one way and then the other.

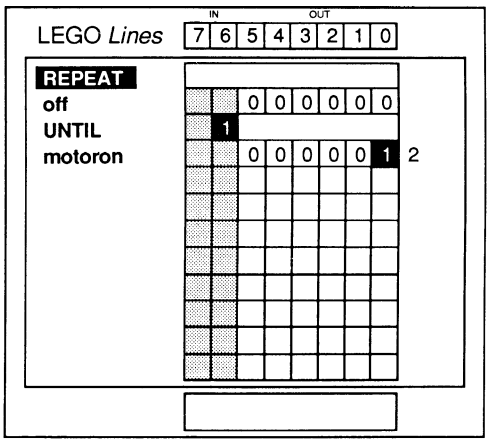
The LEGO ***Lines*** program has a number of reserved words which must not be used as labels because they are words associated with its important decision-making capability. They will always appear in capitals on the screen to distinguish them from labels, which always appear in lower case.

These are: **REPEAT** **UNTIL** **ENDREPEAT** **FOREVER**
IF **ENDIF**
COUNT

Using these reserved words provides structure to a control program.

The following structures are available:

REPEAT UNTIL



Program "SEVEN"

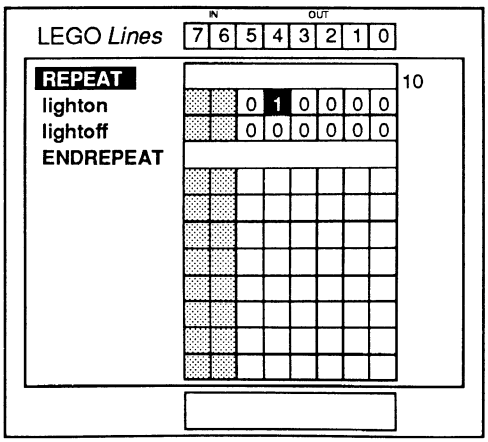
Lines will either be repeated or ignored **UNTIL** a certain input pattern is recognised by the program.

The input pattern on the **UNTIL** line is set using **Apple-6** and **Apple-7**.

This program waits until input bit 6 is set on before it switches on the motor.

Notice that input bit 7 is set to **any value**, meaning that the **UNTIL** line will ignore the state of this particular bit and only look at input bit 6.

REPEAT ENDREPEAT

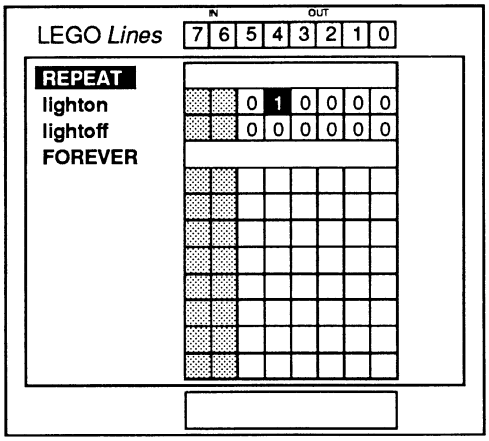


Program "SIX"

Lines between these two reserved words will be repeated a number of times. **ENDREPEAT** marks the end of the sequence of lines to be repeated. The number in the right-hand column of the screen on the **REPEAT** line (entered using the number keys) is how many times the **REPEAT** sequence takes place.

This program turns a lamp on and off ten times.

REPEAT FOREVER

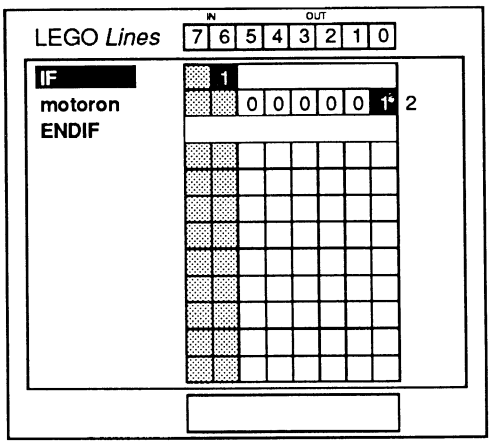


Program "FIVE"

Lines between the **REPEAT** line and the **FOREVER** line will be repeated until the **ESCAPE** key is used to stop the control program.

This program turns a lamp on and off forever, or until the **ESCAPE** key is used to stop it.

IF ENDIF



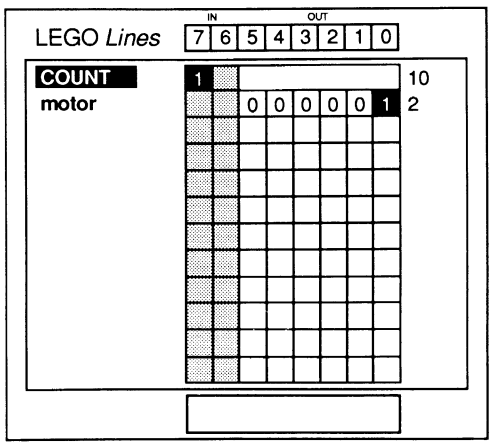
Program "NINE"

Lines will be executed or skipped over according to a certain pattern on the input bits.

The input pattern, on the **IF** line, is set using **Apple-6** and **Apple-7**. **ENDIF** is used to mark the end of the group of lines in question.

This program turns on a motor if input bit 6 is set on.

COUNT



Program "COUNT"

This reserved word is used to count input signals on a single input bit before it goes on to execute the lines which follow. The number of input signals it is counting is entered on the right hand side of the **COUNT** line.

COUNT uses only one of the two input lines (7 or 6). The other input line automatically is set to **any value**.

This program turns on a motor if input bit 7 is set on 10 times.

Assignments 2 and 5 introduce the use of these reserved words.

'Feedback is important in order to know where you are.'



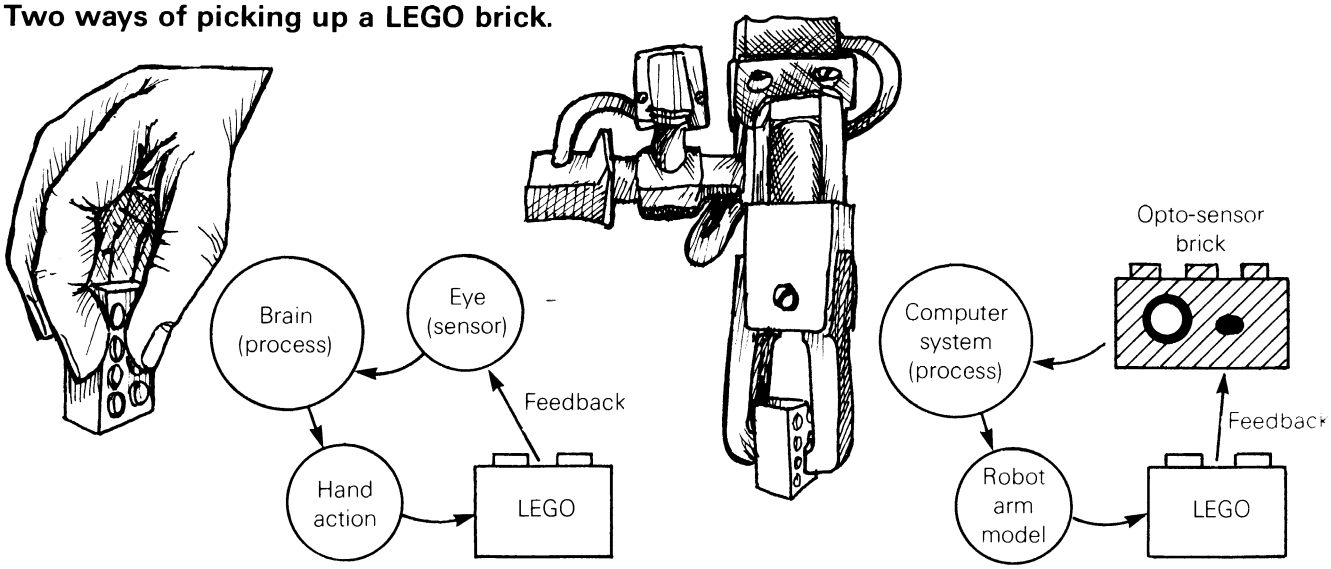
First write down the sentence about feedback.

Now try to do the same thing again, only this time do it *with your eyes shut*.

You were able to write the first sentence better because your eyes sensed where your hand was and fed this information to the brain.

This is called ***feedback***.

Two ways of picking up a LEGO brick.



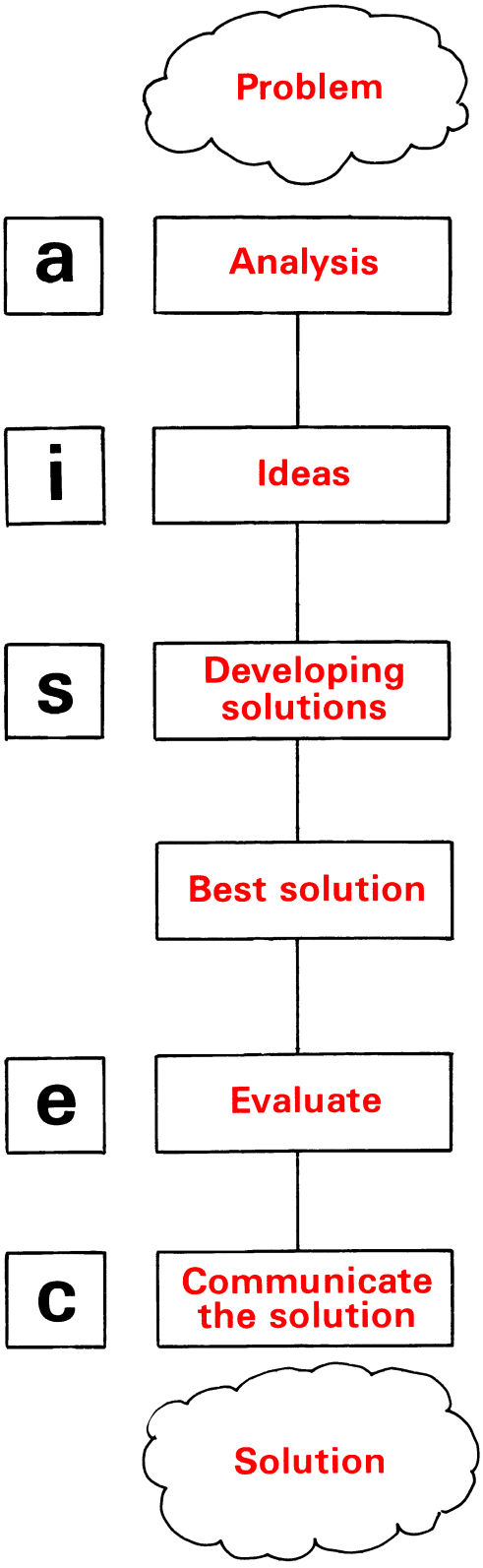
The brain acts on the information it receives from the eye.

The computer system acts on the information it receives from the opto-sensor.

A ***dedicated*** computer system will constantly monitor the feedback from the job it is doing. Frequently, however, computer systems are required to do more than one job at the same time — this is like constantly opening and closing your eyes while you do something. Computer systems usually operate at faster speeds compared to devices they are controlling, and so they are capable of processing instructions to control more than one task.

A program is a sequence of instructions which solves a particular problem. The best test of a program is whether it works.

Other important criteria are whether it is efficient in terms of statements and speed, and whether it can be easily understood and followed.



Express the problem as a sentence or sentences in English. The clearer the statement of the problem the neater the solution will be. Determine also what the solution is required to do.

Identify **Actions** (and give them a meaningful label)

Tests — when actions should or should not take place

Loops — when a sequence of tests and/or actions are repeated.

Use a structured flow diagram (Resources booklet **R14**) to produce solutions.

Choose the best solution and translate the diagram into a program which you can test on the computer.

Judge whether the program achieves what you required it to do.

State what the problem was and describe your solution.

In order to construct a structured flow diagram it is necessary to take a sentence which describes a solution to a problem and identify:

The ***sequence*** of events

The ***actions*** which make up these events

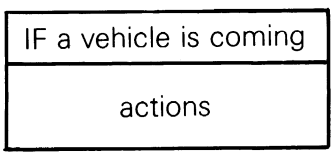
The ***tests*** which decide whether or not those actions take place

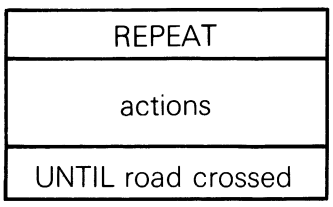
The ***loops*** in which actions or sequences of actions take place.

Look both ways and cross the road slowly if there are no vehicles coming in either direction

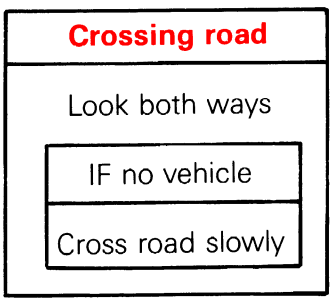
Sequence	Look both ways — if no vehicles — cross the road slowly
Actions	Look both ways — cross the road slowly
Tests	If there are no vehicles coming in either direction
Loops	—

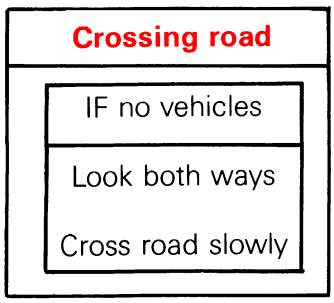
In a structured flow diagram the idea is to place actions into specially shaped boxes



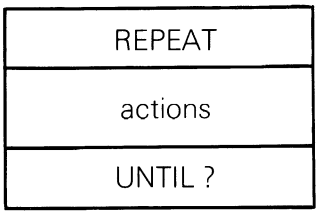


For the example of crossing the road:





Loops



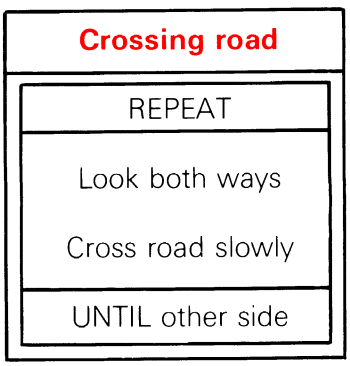
There are a number of ways to organise the condition (?) which finishes the REPEAT/UNTIL loop.

REPEAT UNTIL a certain input message is received.

REPEAT UNTIL a counter reaches a certain value, in other words REPEAT a number of times

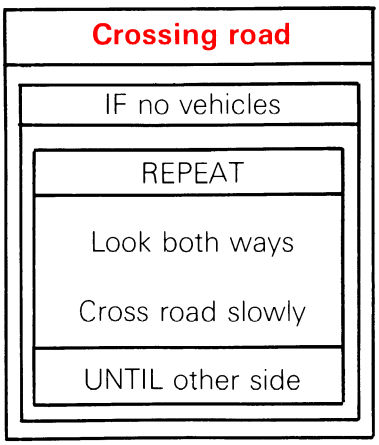
REPEAT FOREVER

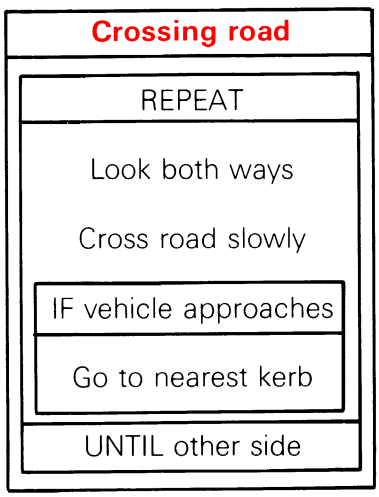
In the example:



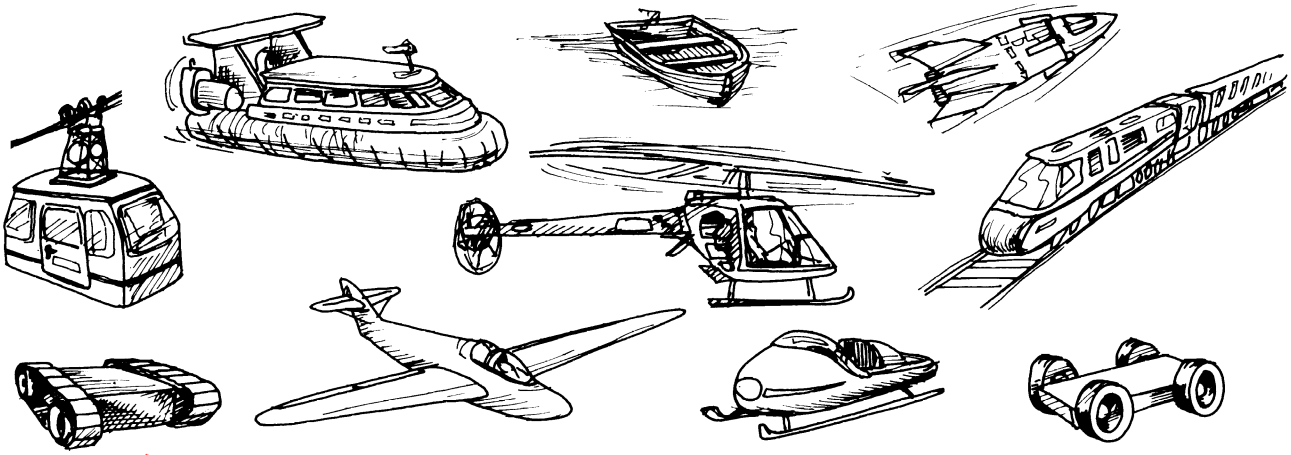
But watch out — a vehicle may be approaching!

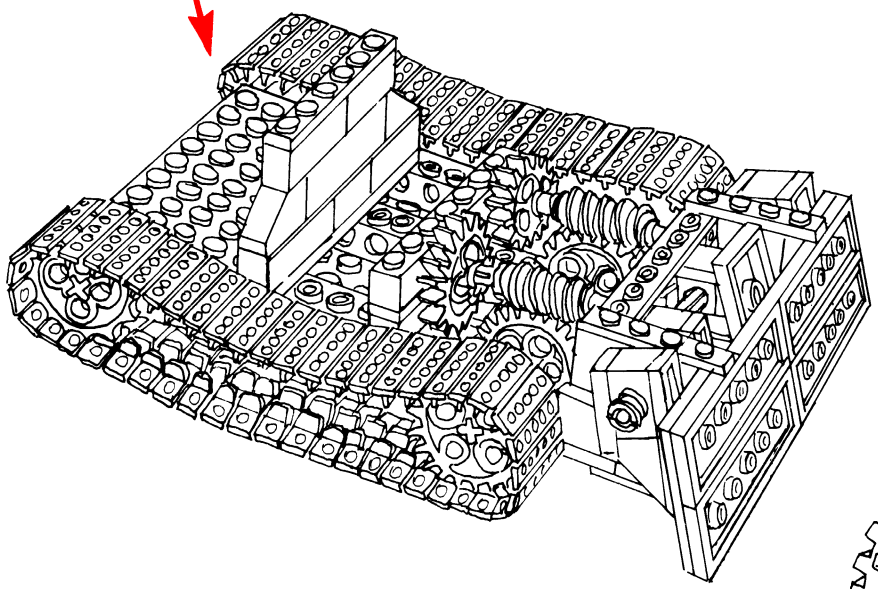
In structured flow diagrams tests and loops may contain each other

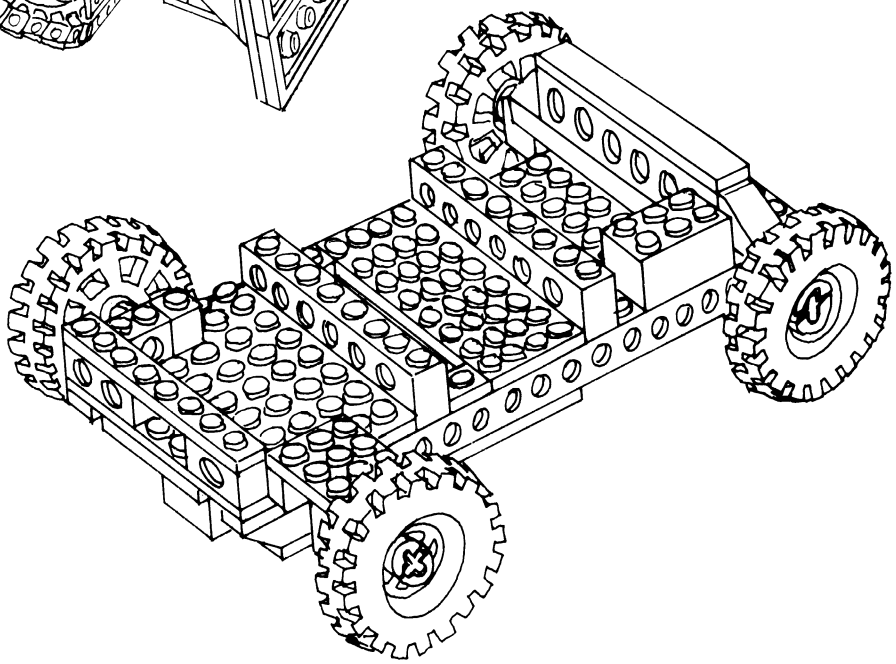


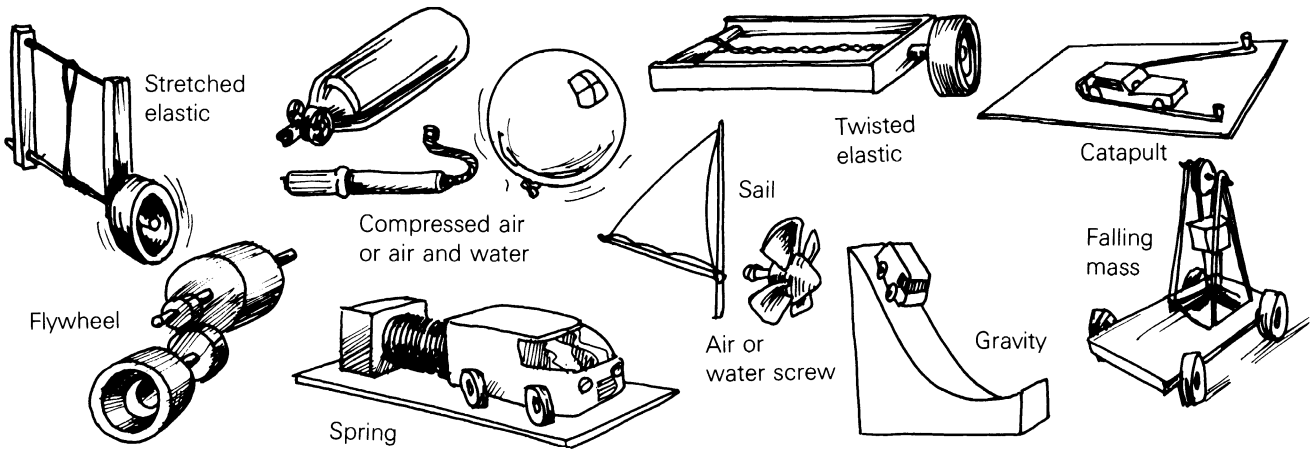


Loops or tests which are within other loops or tests are described as ***nested***.

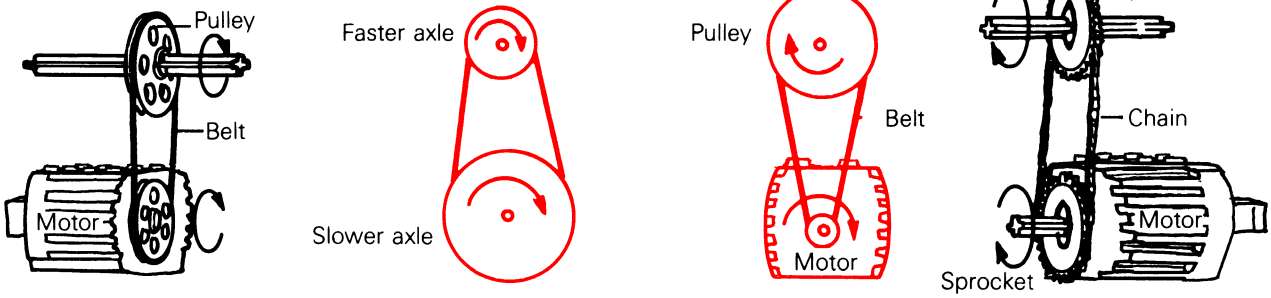




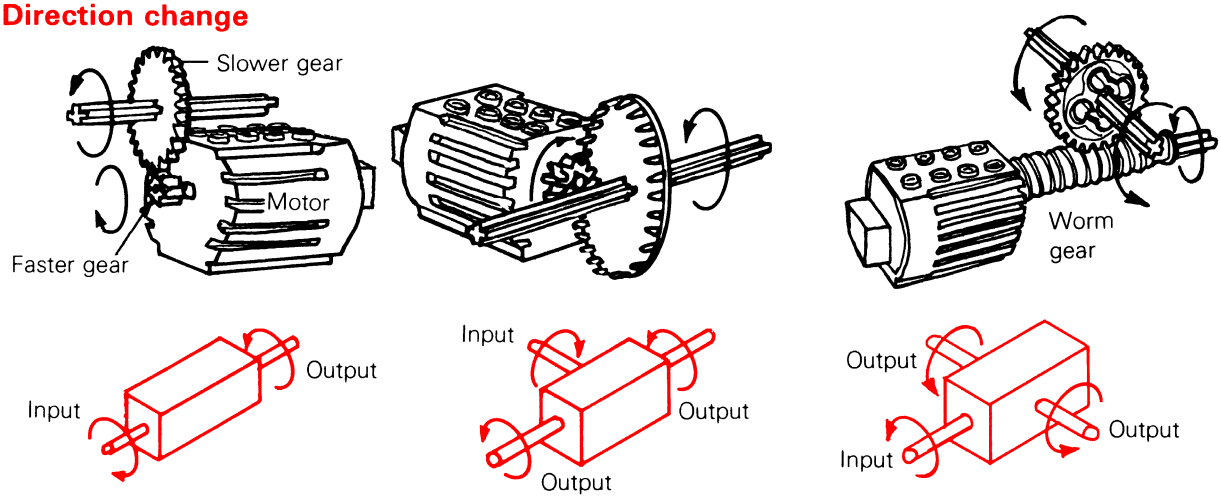




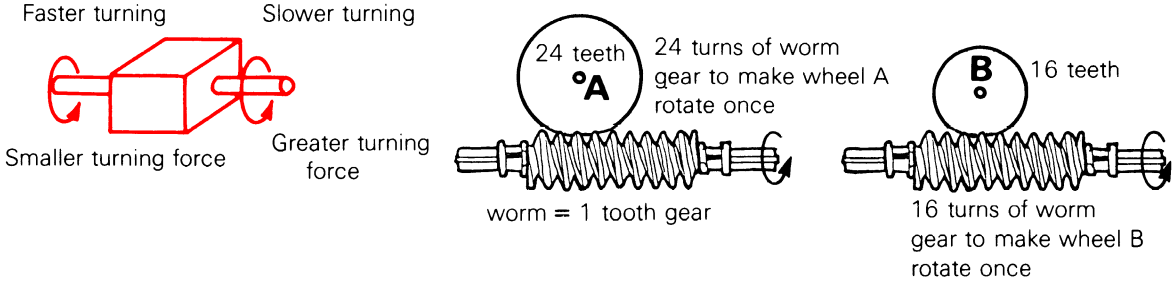
No direction change

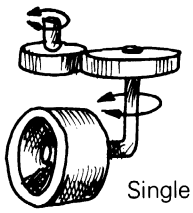


Direction change

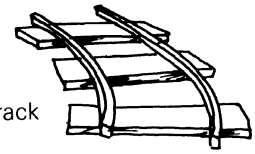


Change of speed

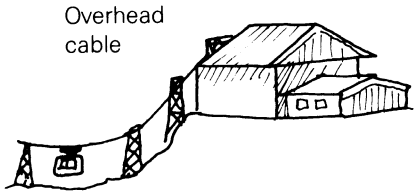




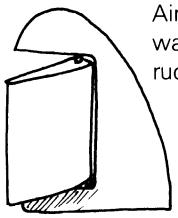
Single wheel



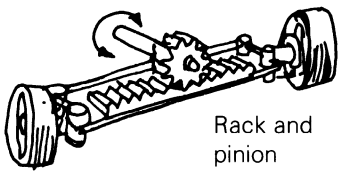
Track



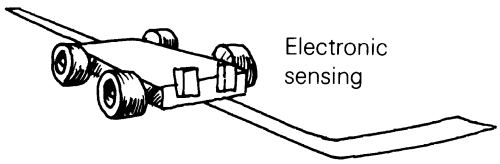
Overhead cable



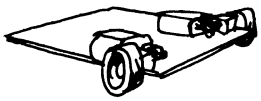
Air or water rudder



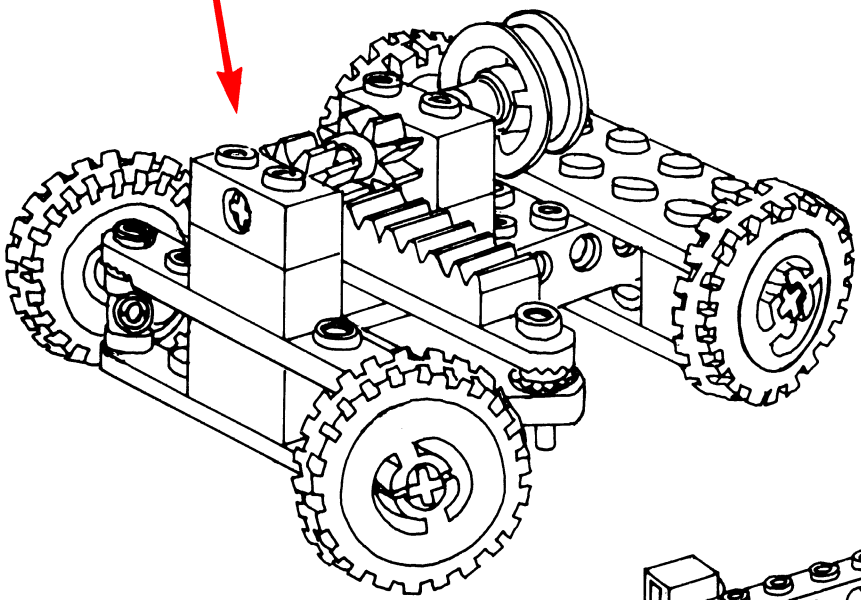
Rack and pinion

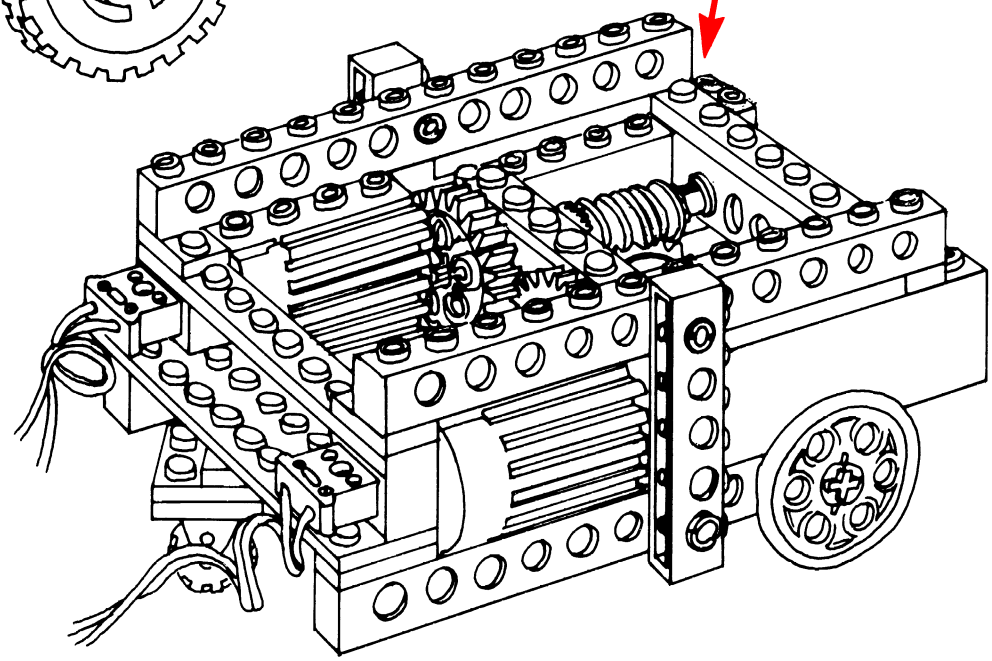


Electronic sensing



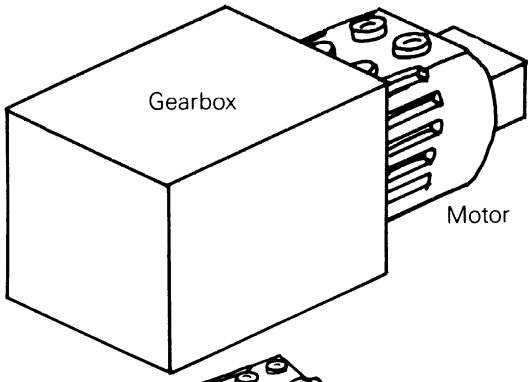
Two-motor control

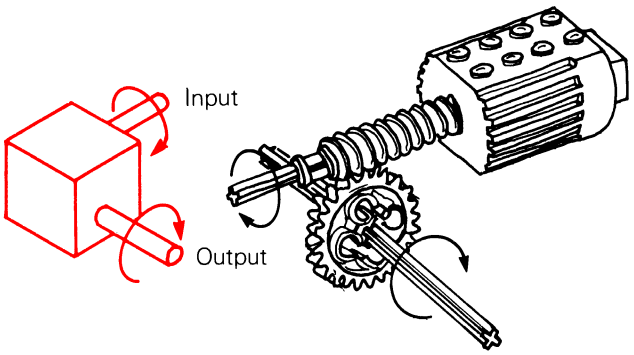


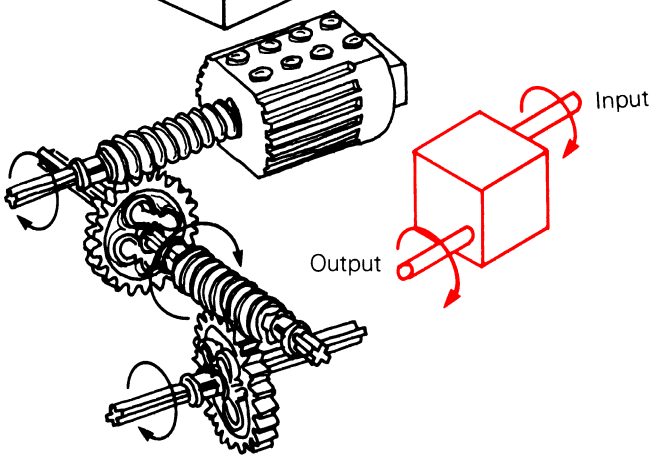


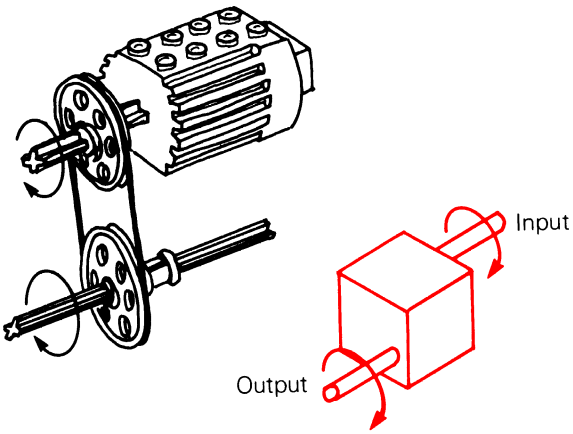
Gearboxes can be used to:

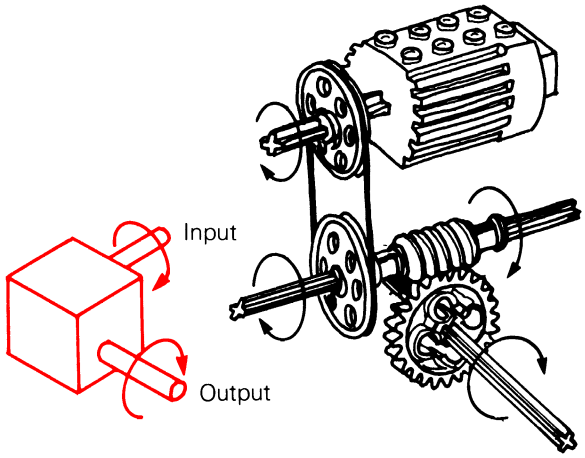
— change speed
— change direction
— change turning force.



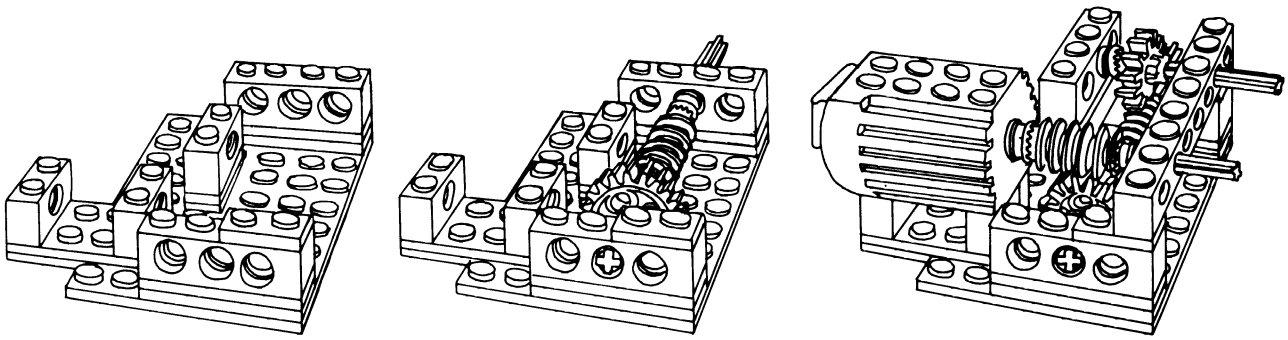


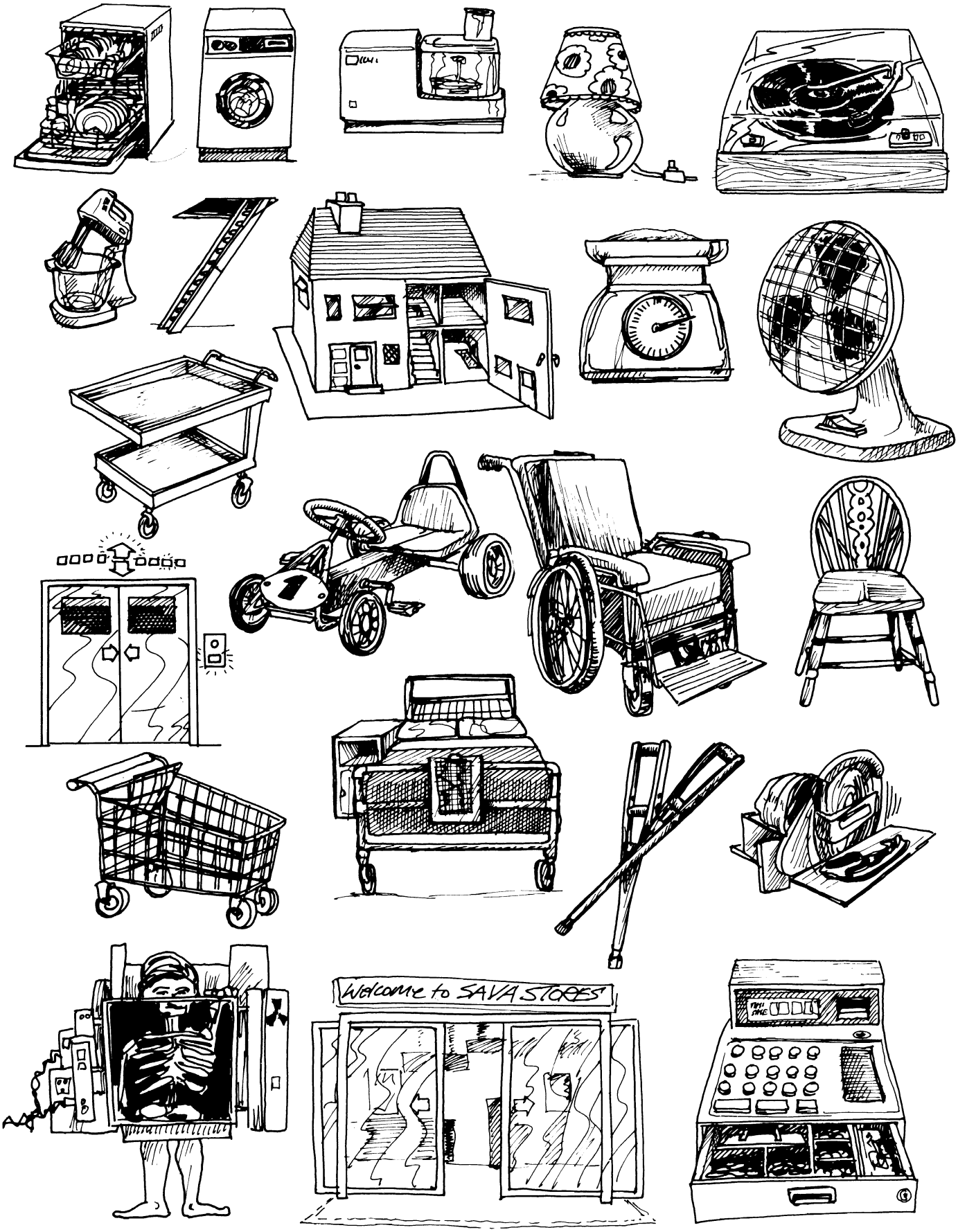


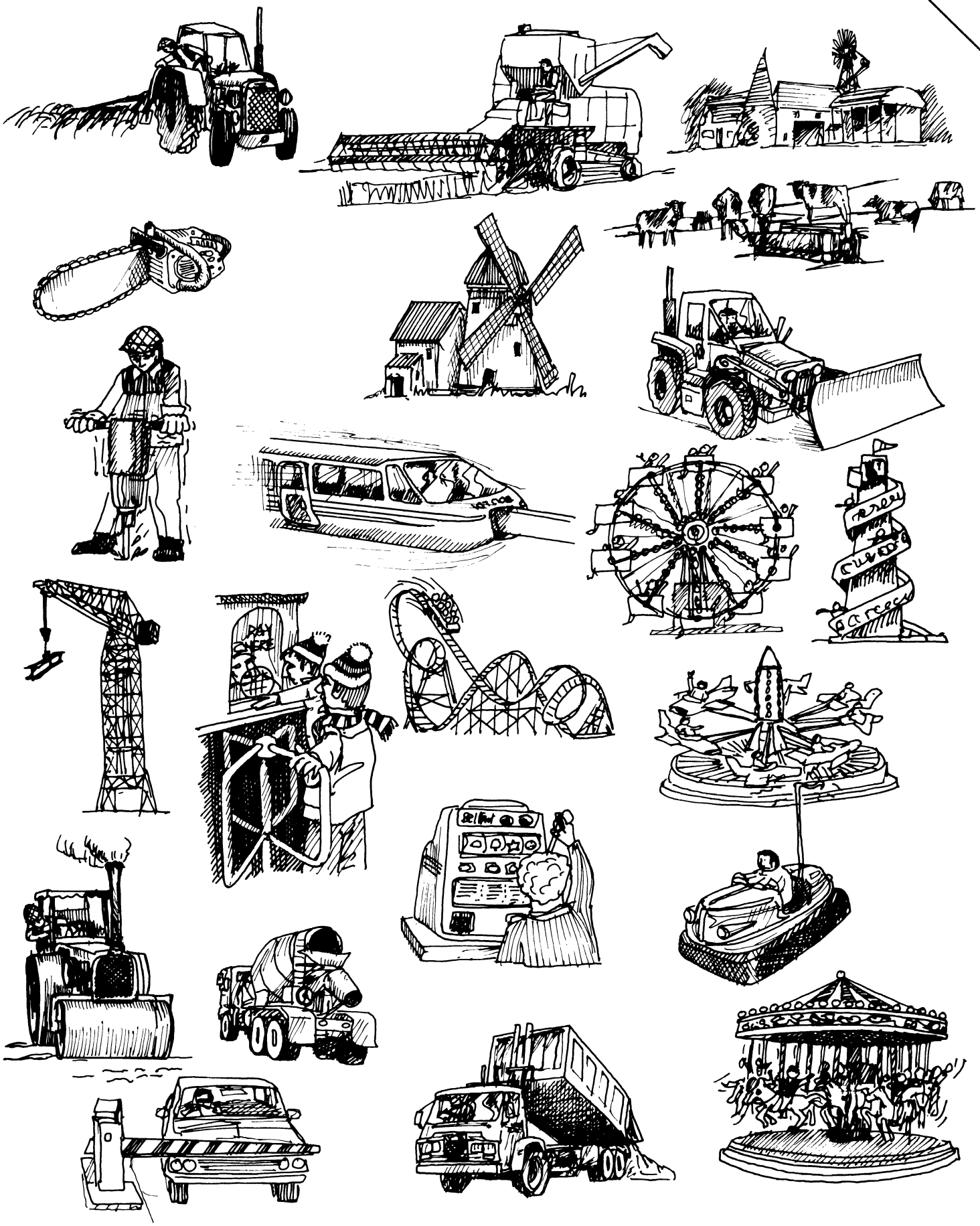


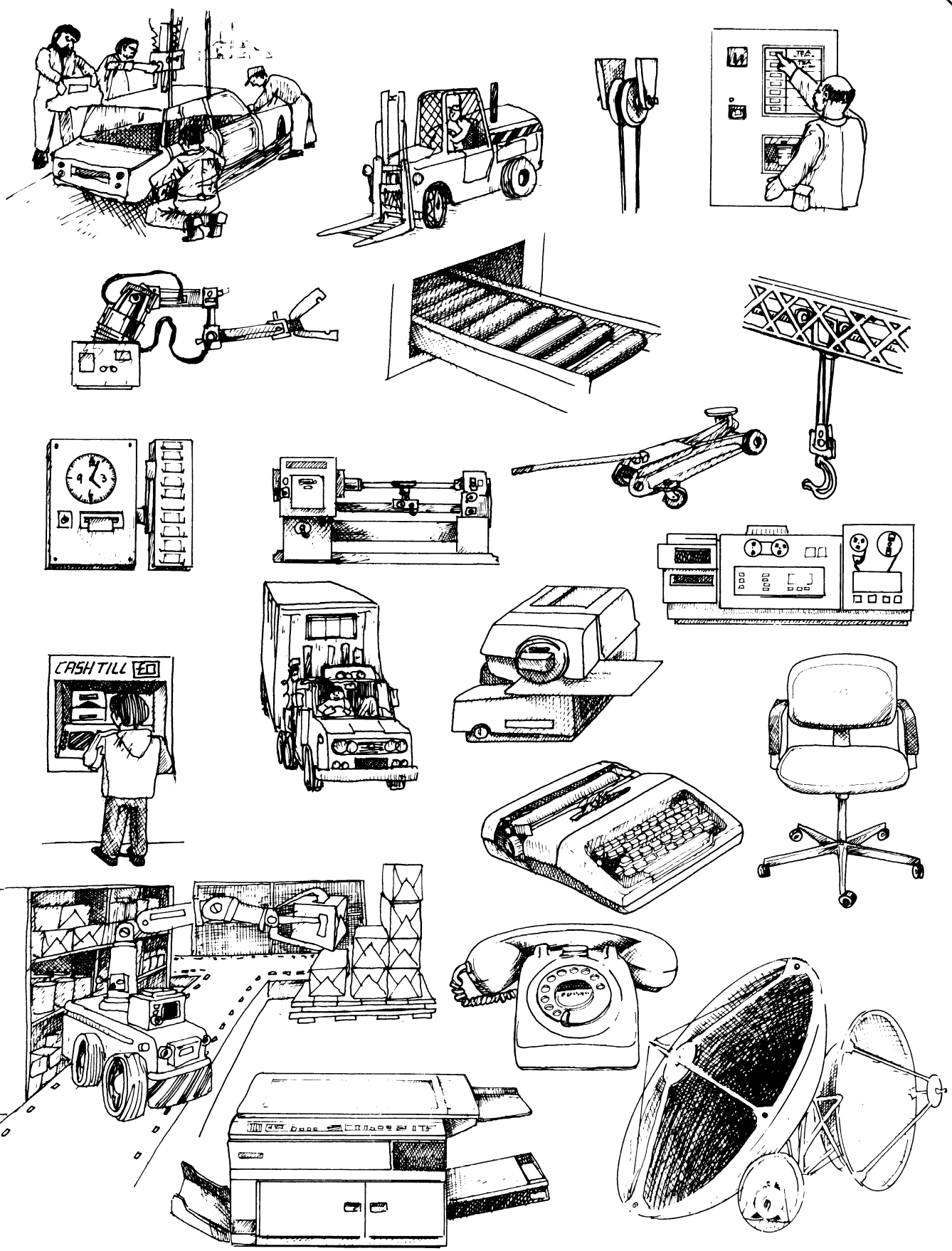


Gearbox for use in Assignment 3




Welcome to SAVA STORES




CASH TILL

Glossary of terms

Algorithm A set or sequence of instructions which describes a solution to a particular problem. **R13 R14**

Any value This term is used in connection with the input bits and describes a state when it doesn't matter whether the input signal from the opto-sensor is ***on*** or ***off***. This only applies when the keywords **IF**, **UNTIL** or **COUNT** are being used to test an input signal on a line. If both input bits are being tested, both cannot be ***any value***: this is an error. If pulses on one input time are being counted the other input line must be set to any value. **R11**

Assignment This is the basic unit of coursework presented in this resource pack. There are ten assignments, **A0 (a–c)** to **A9 (a–j)**. Most have more than one part.

Autoboot A term which is used to describe the automatic loading and running of a program from a disk. To autoboot LEGO ***Lines*** you simply insert the disk into the drive and turn the computer on.**R10**

Bit Short for ***binary digit***. Computers act on signals held in the memory. These signals are combinations of on and off electronic pulses (binary 1 and 0). Because the interface is controlled by a pattern of bits, the connections on the interface itself are referred to as the ***input bits*** (6 and 7) and the ***output bits*** (0–5). **R8 R9**

Closed-loop control In a closed loop, the instructions will be repeated until a new condition is found, at which point the program continues. This condition is tested for either at the start or the finish of the loop, and always involves testing an input signal. **R12**

Commands Software commands enable you to control what the computer does. The autoboot command is an example, as is pressing **Apple-L** to load a program. **R11**

Computer system All the different pieces of computer equipment (monitor, printer, disk drive, cables and leads as well as the computer itself) go to make up the computer system. **R7**

Condition In LEGO ***Lines***, this term describes the pattern of input signals being tested. The pattern you set on the screen is the condition and this is matched against what is happening on the interface. If they are the same, the condition is met and the **IF** or **UNTIL** command is then executed. **R11**

Criteria A list of statements of what the solution to a problem must achieve if it is to be called sucessful. A successful vanilla ice-cream should be white, creamy, sweet and, of course, taste of vanilla. These are all ***criteria***. **R1**

Cursor block The cursor block is the oblong which appears on screen to indicate where the next character will appear when it is typed. It is also used to show which line is being performed when a LEGO ***Lines*** program is being run. **R10**

Drive mechanism This is the means by which the energy from the motor is transferred to the axles, pulleys or gears of a mechanical device. **R16**

Feedback The process by which a controlling system, such as a computer, obtains necessary information about the device it is controlling. **R12**

Gearbox An arrangement of gears which is linked to a motor in order to change the speed, direction or turning force of the output shaft. **R18**

Hard copy Anything which ends up on paper (printout, or just program sheets) which can therefore be saved for future reference, unlike information appearing on screen which is lost. **R10**

Hardware The physical objects making up the system. Printers, monitors, disk drives and so on, are all items of ***hardware***. **R7**

Human senses The five senses (sight, hearing, touch, taste and smell) which provide information which is fed back to the controlling system — the brain — which then acts on this information. **R12**

Indicator lights The red and green lights on the interface. The green (input bits 6 and 7) lights indicate the state of the opto-sensor while the red (output) lights show power being sent to the output bits 0–5. **R7–R9**

Inertia This term refers to the difficulty experienced in making objects change their state; used here in the sense of getting things to move. When you switch a LEGO motor on, there is a very slight delay in moving the other parts attached to it. This is due to ***inertia***, which must be overcome. **R12**

Input bit pattern Connections through which the computer system receives information from the opto-sensors. **R9**

Interface Used to describe the place where two things meet. The term is normally used to refer to a piece of equipment, such as the LEGO interface where signals from the computer meet signals from the outside world. **R7–R9**

Label In LEGO ***Lines***, a label is a word which is entered in the left hand part of the line to describe the action which is taking place (it always appears in lower case). **R11**

LEGO *Lines* A set of specially designed programs to control devices attached to a computer. **R11–R12**

Loop An instruction or a group of instructions which is repeated a set number of times (an open loop) or until a particular condition is met (a closed loop). **R13–R14**

Manual control Where devices are controlled by human beings pressing switches, they are said to be ***manually controlled***. The LEGO manual controller enables you to control models easily. **R6**

Nested loops/tests Loops or tests which are within other loops or tests are described as being *nested*. **R14**

Open loop control Here, the controlling device does not seek any feedback to decide whether to continue the sequence of instructions. **R12**

Opto-sensor brick The LEGO opto-sensor brick contains a sensor which reacts to changes in the level of light which falls upon it. When a change is detected, an electronic signal is sent to the interface. **R9 R12**

Output devices In the LEGO materials, these are the light bricks and motors which may be connected to the manual controller or the output bits of the interface. **R8**

Output bit pattern Connections through which the computer system directs power, to control devices attached to it. **R8**

Printout A copy of any LEGO ***Lines*** program may be obtained on a printer (assuming you have one connected to the computer!) by pressing **Apple-P**. **R7 R10**

Problem-solving process A process, made up of a series of stages, which enables you to approach, and then come to a solution for any type of problem. Here, we are concerned with technological problems. **R1–R5**

Program A sequence of instructions which, when executed in the proper order, enables a computer to solve a problem. Using LEGO ***Lines***, programs may be written to control devices. These programs can be developed and recorded on the program sheets designed for this purpose. **R11 R13**

Reserved words These are the keywords which appear in capital letters when you enter them in the left hand part of the program line. Unlike labels, which merely describe what is happening, these keywords make things happen and are very important in writing efficient programs. **R11**

Simulation When you want to test something, it can be very expensive to build it first and then test it — what happens if it is wrong? (Imagine testing a new aeroplane!) Instead you build a model and test this. Only when you are certain that the real thing is worth building do you go ahead. This is called simulation.

Software This is a set of instructions which makes a computer work. LEGO ***Lines*** (the program, not the disk it comes on) is an example of ***software***. **R11**

Structured flow diagram (*sfd*) A diagram which illustrates the way in which the instructions making up the program solve the problem. **R14**

System Any combination of parts which go together to accomplish a goal. The drinks dispenser in Assignment 5 is a system made up of sub-systems, so is the computer itself. **R7**

Value The right hand part of the program line is where you enter values. These are of two kinds. Time values (in seconds and tenths of seconds) control how long the instruction on that line will be performed for before the program moves on. Number values (whole numbers only) control how many times a loop is repeated or how many input signals are to be counted by the **COUNT** keyword. **R10 R11**

Section 4

Technical reference

Hardware

Required: Apple® IIe or Apple® IIGS with at least 128K RAM and one disk drive.

Optional: Printer with interface in slot 1
AppleMouse® with interface in slot 4
Second disk drive

LEGO: LEGO Slot Card in slot 2 (item 9767)
LEGO Interface Box (item 9750)

Setting up your computer

Before using your Apple computer, you should refer to your Apple® IIe or Apple® IIGS ***Owner's Manual***. Setting up an Apple computer is easy, but if you are not confident, you may wish to enlist some help.

Installing your LEGO Slot Card

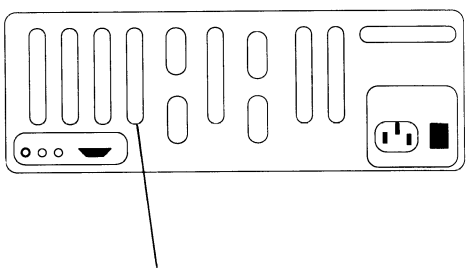
Refer to your Apple® IIe ***Owner's Manual***, Chapter I, for details on installing a peripheral card. To install, the LEGO Slot Card is very much like any standard peripheral card, such as the Apple Disk Controller Card.

First, be sure that the computer has been turned off, and to make things easier and safer, unplug the power cord. Then lift the lid of the computer.

The LEGO Slot Card should be installed in slot 2, so that the chips are on your right, if you are facing the keyboard (front) of the computer. This is the way the other cards are probably facing. Make sure that the LEGO Slot Card is firmly placed.

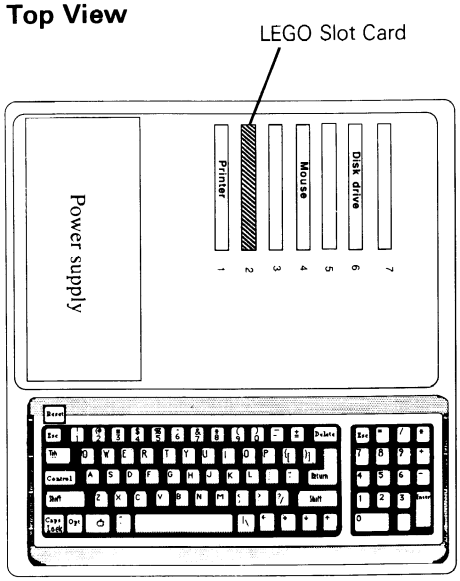
The socket on the connecting cable should be attached to a spare hole in the back of the computer with the screws supplied.

Rear View



The ribbon cable from the LEGO Interface box plugs in here.

Top View



After checking everything, replace the lid, and re-assemble the computer. It is now ready to use.

Using the AppleMouse

Information on using the AppleMouse can be found in the AppleMouse® ***User's Manual*** or in the Apple® IIGS ***User's Manual***. The Mouse may be used to select items from the menu, to select items in the display screen, and to perform certain other tasks.

The AppleMouse Interface must be installed in slot 4 (this is the usual slot for the AppleMouse).

Using a printer

A printer may be used to get a printout of your LEGO ***Lines*** program. Refer to your manuals for your particular printer and your particular printer interface for details on installing and using your printer.

Your printer interface should be in slot 1.

Using an Apple® IIGS

In most respects, using an Apple IIGS is just like using an Apple IIe. However, the following points should be noted:

(a) It is not normally necessary to worry about the location of your AppleMouse interface or your printer interface. They are built into the Apple IIGS in slots 4 and 1 respectively.

(b) It is still necessary to install your LEGO Slot Card correctly into slot 2. In addition, it is necessary to inform the Apple® IIGS that you are using your own card via the Control Panel. See the Apple® IIGS ***Owner's Manual*** (Appendix A, especially page 122) for details. Do this ***before*** starting up the LEGO ***Lines*** disk.

(c) LEGO ***Lines*** will automatically take advantage of the extra speed of the Apple IIGS without affecting the time settings of your programs. However, if you enter the Control Panel while running LEGO ***Lines*** the system may revert to the slower ('normal') speed.

(d) If the Slot Card is put into any other slot apart from slot 2 you will need to change the slot reference on the Setup mode of the LEGO ***Lines*** disk (see page 4.11).

(e) If you are using an RGB monitor you may see a colour effect which may obscure clear reading of the menu bar.

This is a characteristic of Apple IIe graphics which the IIGS is emulating in LEGO ***Lines*** mode.

Software

The LEGO *Lines* program comes on a single disk, with a duplicate on the second side. The program operates under the PRODOS operating system. The files on the disk are as follows:

PRODOS.SYSTEM	BASIC.SYSTEM	STARTUP
LINES	SAGE	FONT.LINES
LINES.SCREEN	LINES.DATA	LINES.LOGOII
LINES.SETUP	LINES.DIR	LINES.SAMPLE
LINES.SCREENØ	ZAP	

A LEGO *Lines* program disk must have all of the above files (except for **LINES.LOGOII** and **LINES.SAMPLE**), or it will not run. The **LINES.DIR** directory on side A contains sample programs, while on side B it is empty. For convenience, the user may duplicate either side to make a fresh LEGO *Lines* program disk.

Preparing disks

Before using the disk, you should make BACKUP copies, and store the original in a safe place. **The original disk will not operate correctly on all functions, but the backups will.**

If you wish to make a copy of the program disk complete with sample files, you should copy side A of your original. If you wish to have a copy without sample files, giving you more room for your own files, you should copy side B.

Duplicating 5.25" disks

To duplicate your 5.25" disk, refer to the Apple® II *Utilities Guide* for details on how to use the Apple II PRODOS System Utilities. You will need to have a blank disk ready, and select the **Duplicate a Disk** option from your menu. It is more convenient if you have two disk drives to perform this task.

Using 3.5" disks

It is also possible to back-up your *Lines* disk from a 5.25" disk onto a 3.5" disk.

To do this, use the Apple II PRODOS utilities to format a blank 3.5" disk. Name the new volume **/LEGO**.

Next, use the **Copy Files** option to copy ALL of the files across to the new disk.

Using LEGO *Lines*

The LEGO ***Lines*** program is a control program that uses the following files:

PRODOS.SYSTEM	Required by system
BASIC.SYSTEM	Required by system
STARTUP	Starts the rest of the program
LINES	The control program
SAGE	The sound and graphics utility
FONT.LINES	The display text
LINES.SCREEN	Display screen
LINES.DATA	Data for LEGO ***Lines*** program
LINES.DIR	Sub-directory containing LEGO ***Lines*** programs
LINES.SETUP	Information on your options
ZAP	Required by system
LINES.SCREENØ	Introductory screen

Starting LEGO *Lines*

LEGO ***Lines*** is designed to run automatically when the disk system is started. There are two ways to start the disk:

With the computer off — Place the disk in drive 1 (label up and outwards)
Turn the computer and monitor on
Close the disk drive door

With the computer on — Place the disk in drive 1 (label up and outwards)
Close the disk drive door
Press **CONTROL Apple-Reset** together (lift off **Apple** last)

After a few moments, the computer will display the main editing screen.

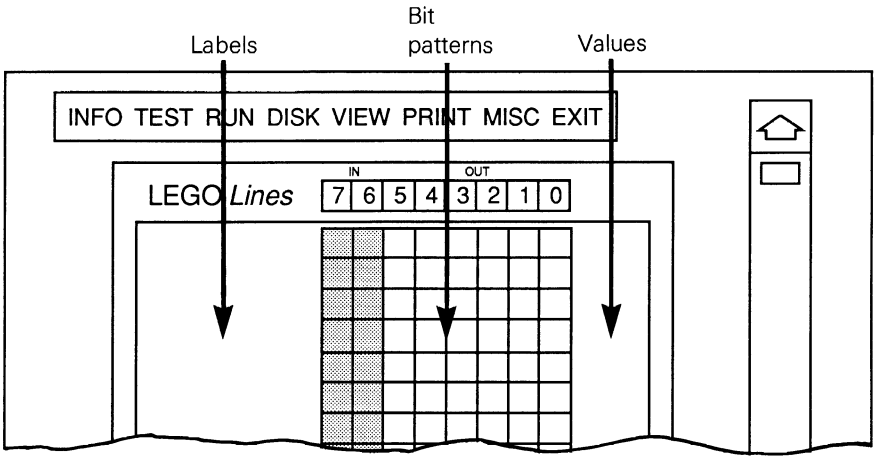
Terminology

In describing how to use the LEGO ***Lines*** program, the following terms and notation will be used:

Using keys:	**PRESS**	Press the key and then release it
	HOLD	Press the key and keep holding it down
	Apple-Key (for example: **Apple-X**)	Hold the ♉ key Press the second key, then release it Release the ♉ key
Mouse:	**MOVE**	Move the Mouse without clicking the button
	CLICK	Click the Mouse button once
	DOUBLE-CLICK	Click the Mouse button twice quickly
	DRAG (for example: Scrolling)	Hold the Mouse button down Move the Mouse Release the Mouse button

The Apple LEGO *Lines* screen

The LEGO ***Lines*** screen is divided into the following areas:



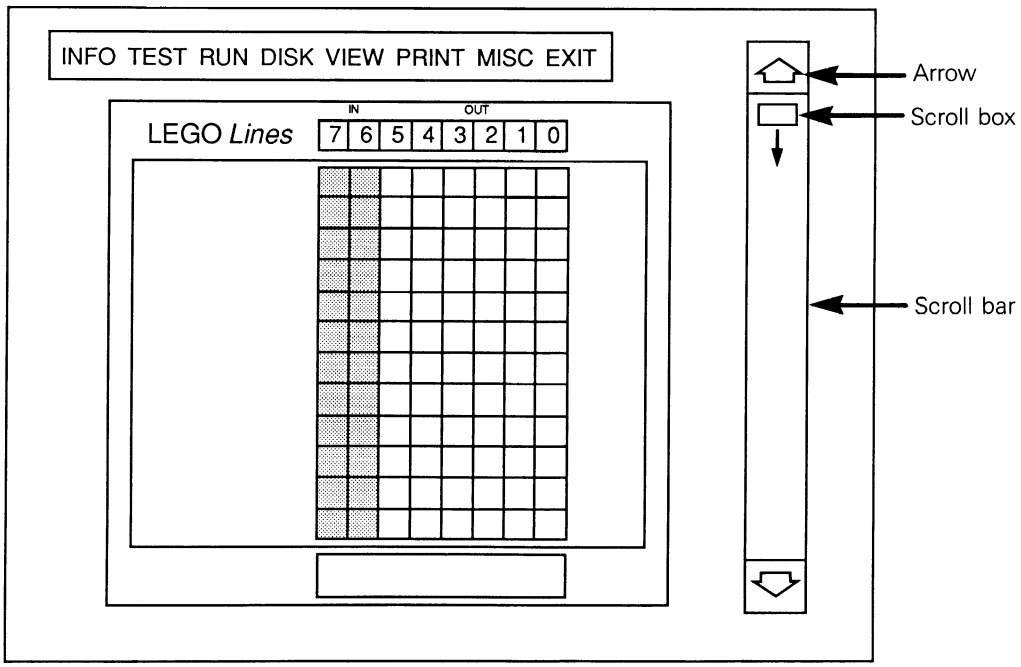
Entering data

LEGO ***Lines*** data is entered into one of three areas, the LABELS area, the BIT PATTERN area, and the VALUES area.

Label or Value data will be entered at the black cursor block. To move to another line, or to another column, use the arrow keys.

Mouse users may select another line or column by moving the Mouse to the desired place and clicking. Another line may also be selected by clicking on one of the arrows on the scroll bar on the right of the screen, or by dragging the scroll box.

It is not possible to move the cursor more than one line beyond the end of your existing program.



Labels

To enter label data (label or keyword), use the 26 letter keys. The key will automatically be displayed in lower-case. No spaces or other characters will be accepted. Typing may be corrected by pressing the **DEL** key to delete letters, or **Apple-B** to clear the whole line.

If the label is recognised as one of the seven keywords, it will automatically be converted to upper-case, and indented to the left. It will not accept any further letters unless it is deleted with the **DEL** key or **Apple-B** key.

If a numeral key is pressed, the cursor will move into the Values column on that line.

Values

To enter value data, use the ten numeral keys or the full-stop or comma key as a decimal point. No spaces or other characters will be accepted. Typing may be corrected by pressing the **DEL** key to delete digits, or **Apple-B** to clear the whole line.

If a letter key is pressed, the cursor will move into the labels column on that line.

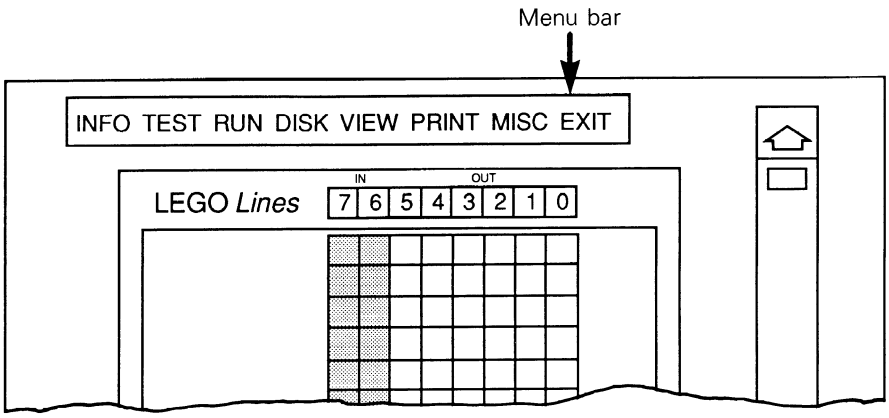
Bit patterns

To enter bit data, hold the **Apple** key and press one of the numeral keys 0-7 where appropriate. Bits 0-5 will alternate between **0** and **1**, while bits 6 and 7 will rotate between **0**, **1** and **any value**. The cursor will not change columns.

Mouse users may use the Mouse to click on the appropriate bit. If the bit is on a different line to the last one used, the cursor will move to that line, without changing columns.

Special functions

Apart from entering program data, the LEGO ***Lines*** program has a number of functions. Most of these functions are available from the menu bar at the top of the screen.



The menu contains the following options:

INFO	Information and Help
TEST	Test a program line
RUN	Run a whole program or part of a program
DISK	View disk contents, Save a program, Load a program, Delete a program
VIEW	View the structure of a program
PRINT	Print the program with structure, show printout on the screen
MISC	Other options: Erase a program, and Setup the system
EXIT	Exit the LEGO ***Lines*** program.

There are three ways to use the functions:

Using the ESC key with the menu

1 Press the **ESC** key. One of the options will be highlighted.
2 Use the left and right arrow keys to select the desired option.
3 Press **RETURN** on the desired option.

Some options, such as **DISK**, have further options. To choose one of these:

4 Use the up and down arrow keys to select the option.
5 Press **RETURN** on the desired option.

To cancel the process, press the **ESC** key.

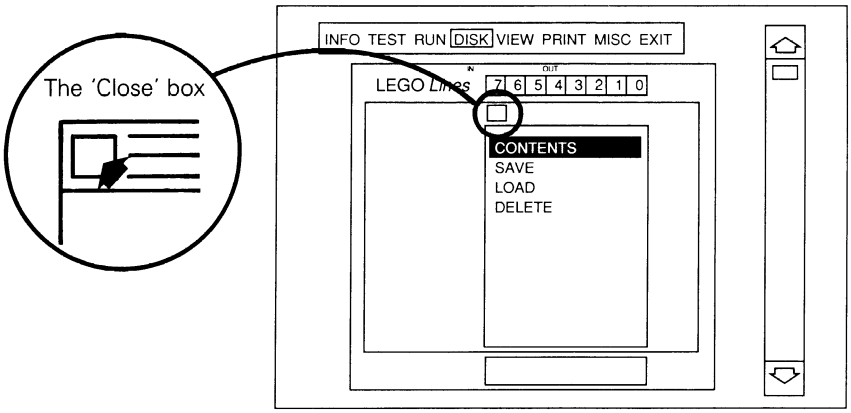
Using the AppleMouse with the menu

1 Move the Mouse to the desired option.
2 Click the button.

Some options, such as **DISK** have further options. To choose one of these:

3 Move the Mouse to the desired option and click the button.
4 Click in the **RET** box to choose.
Double clicking in step 3 will also choose the option.

To cancel the process, click inside the Close box in the top left-hand corner of the window.



Using the open Apple-Key, ♁

Many of the menu functions, and some not on the menu may be used by holding the **Apple-Key** and pressing one of the following:

Apple-I	Insert a line*	**Apple-S**	Save a program
Apple-D	Delete a line*	**Apple-L**	Load a program
Apple-B	Clear (blank out) a line*	**Apple-V**	View program structure
Apple-↑	Top of program*	**Apple-P**	Print program
Apple-↓	Bottom of program*	**Apple-?** or	
Apple-T	Test line	**Apple-/**	Get help
Apple-R	Run the whole program	**Apple-A**	Information about LEGO *Lines*

The five options marked (*) are not available from the menu.

Editing functions

The following functions are used while editing:

INSERT — This inserts a blank line at the current cursor position. The rest of the program, including the current line, will be pushed downwards. If there are the maximum of 30 lines in the program, a line cannot be inserted, and the command will be ignored.

DELETE — This deletes the line at the current cursor position. The rest of the program will be pushed upwards. If there is nothing to delete, this command will be ignored.

BLANK — This will clear (blank out) the line at the current cursor position. The rest of the program will be unaffected.

TOP and **BOTTOM** — The display will show either the top or the bottom of the program, and the cursor will be moved there.

Info functions

ABOUT LEGO *Lines* — This displays information such as the author and the current version number.

HELP — A list of the special keys listed above will be displayed.

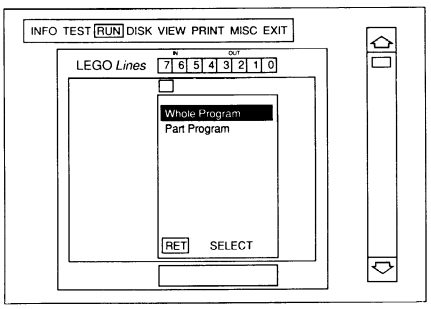
Test and Run functions

TEST — This will test the line at the current cursor position, by sending the bit data to the LEGO Interface as long as either the **Apple** key or the Mouse button is held down.

The value of the current line will be ignored for testing purposes. A keyword line cannot be tested.

RUN — This will run either the whole program or part of the program. If you select **Whole Program** the program will run from the beginning.

If you select **Part Program** the section will start from your current cursor position. You will then be instructed to select the rest of the section with the ↑ or ↓ keys. Be sure not to break up any structures or to include incorrect lines in your selection.



Before running, the program will be tested for structural errors, and these will be reported. The program will not run if such errors exist.

Stopping the program

While running you can stop the program by pressing the **ESC** key. The cursor will then return to the first line. **Apple-R** will restart the program.

Pausing the program

You may also 'pause' a program by pressing the space bar once. The computer will 'click' while the program is 'pausing'. You can press **ESC** to stop the program or press the space bar again to continue the program.

Disk functions

LEGO ***Lines*** is designed to be used in an Apple® IIe or Apple® IIGS with a single disk drive. However, it is more convenient when making copies to use a second disk drive. A second disk drive will not be used for running the program.

LEGO ***Lines*** will also allow the user to save LEGO ***Lines*** program files onto the program disk, to load them, to delete them, and to view the contents of the disk.

To use the disk functions, select **DISK** from the menu. Then choose one of the four options: **CONTENTS**, **SAVE**, **LOAD** or **DELETE**. If you are saving or loading, you may use the **Apple-S** or **Apple-L** commands instead.

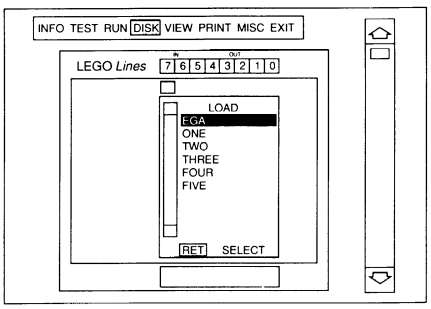
CONTENTS — A list of LEGO ***Lines*** program files will be displayed. If there are no files, a message will appear.

If there are more files than the display can fit, you may scroll through the list by pressing the ↓ or ↑ arrows, or by clicking the Mouse in one of the arrow symbols.

SAVE — The program will be saved onto disk. If there is nothing to save, a message will appear.

The user will be asked for a name. Up to ten characters may be used, with no spaces. The first character must be a letter while the rest may be letters, numerals or full-stops/period.

If a file of that name already exists on the disk, the user will be given the option of over-writing the old file. The file will then be saved.



LOAD — The user may choose from one of the files on the disk to be loaded into memory. First, a list of LEGO ***Lines*** program files will be displayed. If there are no files, a message will appear.

Select the file by pressing one of the arrow keys and then pressing **RETURN**. Mouse users may click on the file name, and then click in the **RET** box. Or they may double-click on the file name.

If there are more files than the display can fit, you may first scroll through the list by pressing the ↓ or ↑ arrows, or by clicking the Mouse in one of the arrow symbols.

If there is already a program in memory, you will be asked whether you wish to erase it. If you press **Y** (yes), the operation will proceed. Otherwise you will be returned to the editing screen.

DELETE — The user may choose from one of the files on the disk to be deleted from the disk. First, a list of LEGO ***Lines*** program files will be displayed. If there are no files, a message will appear.

Select the file by pressing one of the arrow keys and then pressing **RETURN**. Mouse users may click on the file name, and then click in the **RET** box, or double-click on the file name.

If there are more files than the display can fit, you may first scroll through the list by pressing the ↓ or ↑ arrows, or by clicking the Mouse in one of the arrow symbols.

Note that once a file is deleted, it cannot be recovered, so use the option judiciously.

View function

A ***Lines*** program may use a number of structures, such as **REPEAT** or **IF** structures. The **VIEW** function allows the user to see how the program is structured, and whether the structuring is correct.

Before viewing the structures, LEGO ***Lines*** will check for errors, and display them. The program will then be represented as a series of lines. Each label is shown by a short horizontal line.

Structures are shown by a loop, joining the **REPEAT** or **IF** to its corresponding ending. If there is an incomplete structure, it will be shown by a broken loop.

Once the display is complete, the user may return to the EDIT screen by pressing **RETURN**.

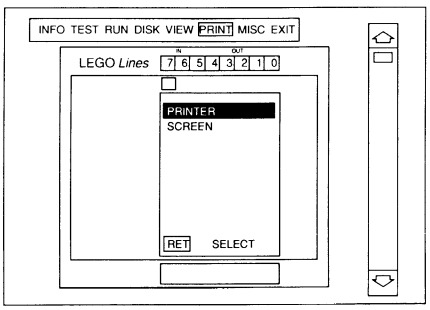
Print function

This will print the LEGO ***Lines*** program, complete with structure lines, to the printer. When this function is selected, you will first be given the choice between printing to the screen, or to the printer. Before selecting the printer, take the opportunity to ready the printer, before either pressing **RETURN** or clicking the Mouse in the **RET** box.

WARNING

If there is no printer, or if it is not ready, the program will 'hang' if you proceed. It is a good idea to check the printer well before planning to use this option.

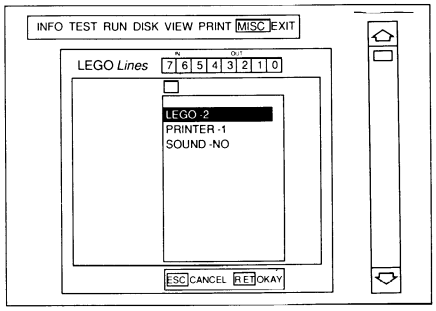
If you change your mind, you may press **ESC** to cancel before printing starts, or while printing is in operation.



Miscellaneous functions

There are two miscellaneous functions. **NEW** will erase the program in memory. **SETUP** will give you the opportunity to change a few options.

NEW — The program in memory may be erased with this option. It will not affect the copy that is still on the disk. **A program that has been erased cannot be recovered, so make sure that you are serious about doing this.**



SETUP — You may set up some of the options for running the LEGO ***Lines*** system. There are three options:

PRINTER SLOT, **LEGO SLOT** and **SOUND** — Under normal circumstances, you will not need to set anything up, so this is here if your circumstances are not so normal. To choose which option to set up, you may select which option with the ↓ and ↑ keys. Then select the value by pressing the ← and → keys.

PRINTER SLOT (Usually slot 1) — This may be anything from slot 0 to slot 7. Slot 0 will display to the screen rather than a printer.

LEGO SLOT (Usually slot 2) — This may be from slot 1 to slot 7.

SOUND (Usually off) — This may be on or off. Some sound such as key click or pausing, will be heard even with sound off.

Exit function

When the user has finished the session, the program may be quit with this option. Be sure you save your current program if you wish to keep it before exiting.

Programming the LEGO Interface

The Apple LEGO system consists of the following components:

An Apple computer to control the system
An Apple LEGO Slot Card in slot 2
A LEGO Interface
A LEGO model

The model is controlled by switching signals from the LEGO Interface. This is in turn controlled by the Apple computer through the LEGO Slot Card.

The LEGO ***Lines*** control program is one program which can control the model through the interfaces. LEGO ***Lines*** is written in Applesoft® Basic, although it makes much use of a machine language sound and graphics utility called SAGE.

It is also possible for the user to write other software to control the model, given a good knowledge of a language such as Applesoft.

The Slot Card

The Apple LEGO Slot Card is based on a Mostek 6522 VIA chip. The chip is communicated with through the I/O addresses, calculated as follows:

```
L = 49280 + S*16
```

where **S** is the slot number, and **L** the resulting address.

All input/output will come through the address **L**, although the next three addresses are used during setup. A typical setup sequence runs as follows:

```
POKE L+3,1
POKE L+2,63
POKE L+1,Ø
POKE L,Ø
```

This sets up the 6522 registers so that bits 0-5 are output bits, and bits 6 and 7 are input bits. All I/O is then done through address **L**.

Programming in AppleSoft®

Note: When entering program listings be sure to distinguish between:

1 (one) and **L** (letter) and between
Ø (zero) and **O** (letter)

Below is a set of Applesoft subroutines which will enable you to make use of the LEGO Interface. They will perform some of the functions of the LEGO ***Lines*** program, although they may be less efficient than the machine code routines often used by LEGO ***Lines***.

These routines assume that the LEGO Slot Card has been installed in slot 2.

Initialising

This routine must be called first to set up the LEGO Interface correctly:

```
1000 REM INITIALISE INTERFACE
1001 S=5:L=49280+S*16
1002 POKE L+3,1
1003 POKE L+2,63
1004 POKE L+1,0
1005 POKE L,0
1006 RETURN
```

Reading data

Below is a subroutine which will read data from the interface, and store this in an array **DB(..)**. This starts off as a single decimal value, but will here be converted into elements **DB(6)** and **DB(7)**, containing 1's or 0's to represent on or off.

```
1200 REM INPUT DATA
1201 DB=PEEK(L)          :REM Read date
1202 DB(7)=(DB>127)      :REM Convert binary data
1203 DB=DB-128*DB(7)
1204 DB(6)=(DB>63)
1205 RETURN
```

This next routine uses the above to test whether certain bits are on or off, as required. It also allows you to set either test bit as **any value**. To use it, you must first set the test bits **T(7)** and **T(6)** (**Temporary 7** and **Temporary 6**) to 0, 1 or -1 (for **any value**). The results will return as **T=1** for true, and **T=0** for false.

```
1210 REM TEST INPUT BITS
1211 GOSUB 1200          :REM Read Input Bits
1212 T=(T(7)=DB(7) OR T(7)=-1) AND (T(6)=DB(6) OR T(6)=-1)
1213 RETURN
```

For example, suppose you wish to test whether bit 7 was on, while bit 6 could be **any value**. Then you could write:

```
T7=1:T6=-1:GOSUB 1210:IF T THEN (etc)
```

Sending data

Below is a subroutine which will send the data stored in an array **DB(..)** to the interface. It is assumed that the elements **DB(Ø)** through **DB(5)** contain 1's and 0's to turn the bits on or off. These data elements are then combined into a single decimal value to be sent.

```
11ØØ REM OUTPUT DATA
11Ø1 DB=Ø                  :REM Initialise data
11Ø2 FOR I=Ø TO 5          :REM Convert binary data
11Ø3 DB=DB+DB(I)*2^I
11Ø4 NEXT I
11Ø5 POKE L,DB             :REM Send data to interface
11Ø6 RETURN
```

The following routine will turn on all the bits specified. It has two entry points. If you call it at the beginning (line 1110), you will turn off all the other bits. If you call it at line 1115, you will leave the other bits alone.

To use this routine, you must first set the required bits **(T(Ø)** to **T(5))** to 1.

```
111Ø REM TURN ON SPECIFIC BITS
1111 FOR I=Ø TO 5          :REM Turn off all bits first
1112 DB(I)=Ø
1113 NEXT I                :REM Falls through to next part

1115 FOR I=Ø TO 5
1116 DB(I)=DB(I) OR T(I)  :REM Turn on required bits
1117 NEXT I
1118 GOSUB 11ØØ            :REM Send data
1119 RETURN
```

For example, to turn on bits 3 and 4, without changing the other bits, you could use the following line:

```
T(3)=1:T(4)=1:GOSUB 1115
```

To turn on bits 3 and 4, and the rest off, use:

```
T(3)=1:T(4)=1:GOSUB 111Ø
```

Finally, the following routine will turn the desired bits on and off for a set period of time, testing for the **ESC** key while waiting. Note that, like LEGO ***Lines***, it will not turn them off at the end of the routine. The data for this routine must be set as in the above routine at line 1115. In addition, the Output Time (**OT**) must be set (to 1 decimal place).

```
112Ø REM TIMED OUTPUT
1121 ES=Ø                  :REM Not ESCaped (yet)
1122 FOR J=1 TO OT*25      :REM OT seconds
1123 GOSUB 1115            :REM Send data
1124 IF PEEK(49152)=155 THEN ES=1:GOTO 1129   :REM ESC
1125 NEXT J
1129 RETURN
```

An example

The following routine emulates the LEGO *Lines* **COUNT** function. To use it, you must first set the Input Bit (**IB**) at 6 or 7 to indicate which bit you wish to test; you must then indicate which Test State (**TS**) you wish to test (On=1 or Off=Ø); finally, you need to set the Count Number (**CN**) to indicate how many times it is to wait for that state.

The routine also tests for the **ESC** key.

```
13ØØ REM COUNT ROUTINE
13Ø1 IF (IB<>6 AND IB<>7) OR CN<1 THEN RETURN   :REM Test Invalid Data
13Ø2 ES=Ø                                       :REM Not ESCaped (yet)
13Ø3 T(13-IB)=-1                                :REM Set other Test Bit to any value
13Ø4 FOR I=1 TO CN                              :REM Count Number
13Ø5 T(IB)=1-TS                                 :REM Set opposite state
13Ø6 GOSUB 121Ø                                 :REM Test Input
13Ø7 IF PEEK(49152)=155 THEN ES=1:GOTO 1319     :REM Test for ESC
13Ø8 IF T=Ø THEN 13Ø6                           :REM Wait
13Ø9 T(IB)=TS                                   :REM Now test for desired state
131Ø GOSUB 121Ø
1311 IF PEEK(49152)= 155 THEN 1319
1312 IF T=Ø THEN 131Ø
1313 NEXT I
1319 RETURN
```

Programming in machine language

Although the LEGO Interface can be programmed in BASIC, some of the LEGO *Lines* program is written in 6502 machine code, to improve performance.

This section is intended only for experienced machine language programmers who wish to write their own routines for controlling the LEGO Interface. It only discusses how to initialise the interface, and how to get data to and from it.

All hexadecimal values are denoted by the prefix **$** (for example, **$2A**), while binary values are denoted with the prefix **%** (for example, **%1Ø11ØØ**).

The address of the interface I/O port is given by the following:

```
LEGO    EQU    $CØ8Ø+$1Ø*S
```

where **S** is the slot number of the LEGO Interface. Normally, where the LEGO Interface is in slot 2, the address is given by:

```
LEGO    EQU    $CØAØ
```

Initialisation

The correct initialisation sequence for the interface is as follows:

```
LEGO    EQU    $CØAØ
INIT    LDA    #1
        STA    $CØA3
        LDA    #$3F
        STA    $CØA2
        LDA    #Ø
        STA    $CØA1
        STA    $CØAØ
        RTS
```

Writing data

All data is written to address **$CØAØ**. The six data bits on the interface correspond exactly to the six data least significant bits written to the address. So, for example, to turn on bits 1 and 3 only (and the rest off), it is necessary to load the accumulator with binary **%ØØ1Ø1Ø ($ØA)**.

Below is the routine to send the bits in the accumulator to the interface:

```
SEND    AND    #$3F      :Mask with %ØØ111111
        STA    $CØAØ     :to filter out bits 7 & 6
        RTS
```

Reading data

Reading the data is more complex. First, the data must be read from the interface. Then the correct bits must be filtered out, if necessary. Finally, the appropriate bits must be examined.

To read the current status of the output bits (to read which bits are currently on):

```
STATUS  LDA    $CØAØ
        AND    #$3F      :Filter out bits 7 & 6
        RTS

TEST    JSR    STATUS
        STA    TEMP
        LDA    #Ø
        STA    COUNT
TØ      ROR               :Rotate last bit into Carry
        BCC    T1         :No bit
        JSR    ACT        :Else act on it
T1      INC    COUNT
        LDA    COUNT      :Up to 6?
        CMP    #6
        BNE    TØ
        RTS
```

Programming in LOGO II

The following routines allow you to communicate with the LEGO Interface with the LOGO II language.

Initialisation

To initialise the LEGO Interface, and set the variable LEGO port to the correct address (assuming slot 2), use the following procedure. Writing to the address is accomplished with the deposit command.

```
TO lego_init
make "lego_port 49280+5*16
.deposit :lego_port+3 1
.deposit :lego_port+2 63
.deposit :lego_port+1 0
.deposit :lego_port 0
END
```

Reading data

Reading data, is a matter of reading the lego_port address, and determining whether bits 7 and 6 have been turned on. Reading the address is accomplished with the examine command. The output of the following procedure are **"bit7 and "bit6 (1=on)**.

```
TO lego_in
local :x
make "bit7 0 make "bit6 0
make "x .examine :lego_port
if :x>127 [make "bit7 1]
make "x :x-128* :bit7
if :x>63 [make "bit6 1]
END
```

Writing data

To write data, the bits to be written are first identified. These bits are then converted to a decimal number to be sent to the address.

The syntax for the procedure is:

```
lego_out [list]
```

where list is a list of numbers (from 0 to 5) which correspond to the bits to be turned on. The remaining bits will be turned off. For example, the command:

```
lego_out [1 3 4]
```

will turn on bits 1, 3 and 4, and the rest off.

The procedure makes use of another procedure, **power** in the conversion routine. In the long **IF** line, the exclamation mark at the end of the line indicates that it continues on the next. You should not type this in.

```
TO lego_out :bits
local ["x "byte]
make "byte Ø
if not emptyp :bits [repeat count :bits !
  [make "x first :bits make "bits butfirst :bits !
  make "byte :byte + power 2 ;x]]
.deposit :lego_port :byte
END

TO power :base :index
local :x
if :base=Ø [op Ø]
make "x 1
if or :base=1 :index=Ø [op 1]
repeat :index [make "x :x*base]
END
```

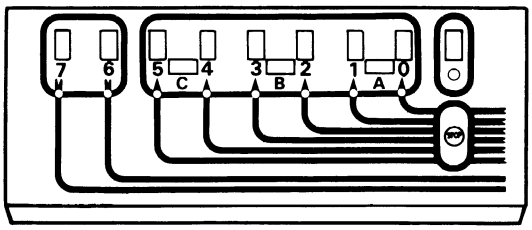
A closing note

The above routines in Applesoft BASIC, 6502 machine language and LOGO II are designed to allow the programmer to experiment further with the LEGO Interface. While they do not form a complete program in themselves, they will allow programmers to communicate with the LEGO Interface in their own programs.

Section 5

Duplication masters

Program sheet



	IN		OUT						
	7	6	5	4	3	2	1	0	

© This material copyright LEGO Group, 1986. ®LEGO is a registered trade mark belonging to LEGO Group. This material may be freely photocopied *for use in the purchasing institution only.*

Assessment



a **Analysis**

Stating the problem accurately
Exploring the problem
Developing deeper understanding
Developing criteria

s **Finding and developing the best solution**

Evaluating each solution
Choosing the best solution
Modelling the best solutions

c **Communication**

Discussion with others
Listening to other's ideas
Drawing sketches and diagrams
Writing a report
Giving a talk

i **Finding and developing ideas**

Discussion with others
Looking and thinking up ideas
Investigating ideas
Finding several solutions

e **Evaluation**

Using the criteria to carry out tests
Analysing the tests' results
Making modifications when necessary

t **Teamwork**

Contributing ideas
Sharing work equally
Working cooperatively

Assignment	*Notes*

Assignment	*Notes*

Assignment	*Notes*

Assignment	*Notes*

Assignment	*Notes*

© This material copyright LEGO Group, 1986. ®LEGO is a registered trade mark belonging to LEGO Group. This material may be freely photocopied *for use in the purchasing institution only.*

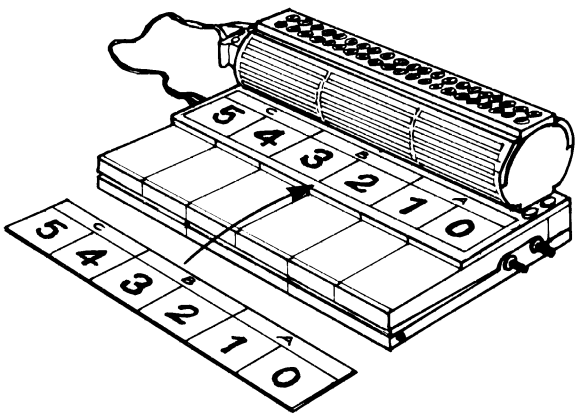
c		b		a © 1986 LEGO Group	
5	4	3	2	1	0

c		b		a © 1986 LEGO Group	
5	4	3	2	1	0

c		b		a © 1986 LEGO Group	
5	4	3	2	1	0

c		b		a © 1986 LEGO Group	
5	4	3	2	1	0

c		b		a © 1986 LEGO Group	
5	4	3	2	1	0



Quick Reference Card

LEGO *Lines* Quick Reference Card

	1 **MOVE ...** the cursor	2 **TEST ...** a line	3 **RUN ...** a program from the beginning	4 **RUN ...** part of a program	5 **DELETE ...** a character	6 **DELETE ...** a line	7 **BLANK ...** out a line	8 **INSERT ...** a line	9 **STOP ...** a program that is running
Keyboard	Arrow keys ↑↓←→ ⌘-↑ to top line ⌘-↓ to bottom line	⌘-T	⌘-R	Position cursor on first line of portion to be run . . . Press **ESCAPE**. Select **RUN** Select **PART** Select portion to be run. Press **RETURN**	Press **DELETE**	⌘-D	⌘-B	⌘-I	Press **ESCAPE**. (⌘-**R** to run again from start)
Mouse	move then click	TEST hold button down	RUN	Position cursor on first line of portion to be run . . . RUN PART Select portion to be run. Press **RETURN**.	—	—	—	—	—

	10 **PAUSE ...** during a program	11 **SAVE ...** a program	12 **LOAD ...** a program	13 **VIEW ...** structure of program	14 **PRINT ...** the program	15 **ERASE ...** a program from memory	16 **DELETE ...** a program from disk	17 **CONTENTS ...** of the disk	18 **HELP ...** with key functions
Keyboard	Press **SPACE** bar to pause Press **SPACE** bar to continue **ESC** to stop	⌘-S	⌘-L	⌘-V	⌘-P	Press **ESCAPE** Select **MISC** Select **NEW**	Press **ESCAPE** Select **DISK** Select **DELETE**	Press **ESCAPE** Select **DISK** Select **CONTENTS**	⌘-?
Mouse	—	DISK SAVE	DISK LOAD	VIEW	PRINT	MISC NEW	DISK DELETE	DISK CONTENTS	INFO HELP



LEGO Australia Pty Limited

EDUCATIONAL MATERIALS DIVISION

2 LINCOLN STREET, LANE COVE 2066

Designed and produced by Tecmedia Limited Loughborough Leicestershire England